Sistema de deslastre de cargas elétricas adaptado para o setor industrial

Tiago José Pastor da Costa
Mestrado Integrado em Engenharia Física
Departamento de Física e Astronomia
2014

Orientador
Professor Paulo Marques (Faculdade de Ciências da Universidade do Porto)

Coorientador
Engenheiro Carlos Santos (EWEN – Energy)

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.
O Presidente do Júri,

Porto, _____/_____/_____

Agradecimentos

Como autor deste relatório, eu gostaria de usar esta oportunidade para agradecer às pessoas que me ajudaram e estiveram sempre presentes neste trabalho.

Aos meus orientadores, Professor Paulo Marques e Engenheiro Carlos Santos pelo apoio, orientação e disponibilidade ao longo do projeto.

À minha família por estar sempre presente, por todo o apoio e incentivo.

Aos meus amigos pelo companheirismo e força ao longo do decorrer de todo o processo e dos anos.

Aos Engenheiros Luís Ribeiro e José Lourenço pela oportunidade de me juntar à EWEN no desenvolvimento deste projeto. Permitindo-me sempre tomar as minhas próprias decisões e disponibilizando sempre todo o apoio necessário.

À minha namorada por nunca perder a paciência, pela compreensão e por nunca me deixar passar pelos momentos mais difíceis sozinho.

"Learning never exhausts the mind."

Leonardo da Vinci

Índice

Índice de figuras

Índice de tabelas

Lista de Abreviaturas

- USB – *Universal Serial Bus*
- SRAM – *Static Random Access Memory*
- EEPROM – *Electrically-Erasable Programmable Read-Only Memory*
- RTC – *Real Time Clock*
- SD Card – *Secure Digital card*
- ICSP – *In-circuit serial programming*
- TCP – *Transmission Control Protocol*
- UDP – *User Datagram Protocol*
- LED – *Light Emitting Diode*
- GCC - *GNU Compiler Collection*
- RAM – *Random Access Memory*
- I2C – *Inter-Integrated Circuit*
- SCL – *Substation Configuration description Language*
- HTTP – *Hypertext Transfer Protocol*
- HTTPs – *HyperText Transfer Protocol Secure*
- FTP – *File Transfer Protocol*
- SMTP – Simple Mail Transfer Protocol
- DNS – Domain Name System
- DHCP – Dynamic Host Configuration Protocol
- TFTP – Trivial File Transfer Protocol
- SNMP – Simple Network Management Protocol
- VOIP – Voice over Internet Protocol
- MODBUS - Serial communications protocol originally published by Modicon

Resumo

No decorrer deste projeto pretendeu-se desenvolver um sistema de monitorização e controlo de sistemas com capacidade para deslastre de cargas elétricas adaptado para ambientes industriais. Todo o desenvolvimento do projeto terá como base um microcontrolador Arduíno apoiado por uma placa de comunicação *Ethernet* que irá permitir a troca de informação entre a rede e o microcontrolador.

O termo deslastre de carga está definida pelo regulamento de operação das redes elétricas como a interrupção da alimentação de alguns consumos de energia elétrica com o objetivo de preservar o funcionamento do sistema elétrico, a nível local ou nacional, em condições aceitáveis de tensão e frequência. (ENERGÉTICOS, 2010)

Recorrendo a este tipo de equipamento pretende-se uma leitura e controlo de parâmetros físicos como temperatura, pressão, fluxo de líquidos, humidade e outros, com o intuito de criar um sistema autónomo com capacidade de recolher dados para a base de dados da EWEN – ENERGY e de enviar comandos de acordo com as especificações do utilizador, como por exemplo o abrir e fechar de uma válvula.

Durante o desenvolvimento do projeto surgiu também a proposta de criação de um contador de impulsos com capacidade para configuração através de uma página web onde são definidos parâmetros como o número de canais ativos e os parâmetros da rede para onde se pretende que sejam enviados os dados recolhidos. O objetivo principal está na aplicação para leituras de consumos energéticos enviados por emissores de impulsos que enviam a informação da energia consumida por equipamentos. Será portanto um novo equipamento, distante do objetivo inicial mas que apresenta capacidades promissoras a serem aplicadas num contexto industrial.

Palavras-chave: deslastre, controlo energético, eficiência energética, parametrização analógica e digital com Arduíno.

Abstract

The purpose of this project was to develop a method of system monitoring and control, capable of load shedding, adapted to industrial environments. The whole development of this project will be based on an Arduino microcontroller supported by an Ethernet communication that will enable the exchange of information between the network and the microcontroller.

Load shedding is, as defined by the regulation of power grids operation, an intentionally engineered electrical power shutdown where electricity delivery is stopped for non-overlapping periods of time over different parts of the distribution region, in order to preserve the functioning of the electrical system at local or national level, in acceptable conditions of voltage and frequency. (ENERGÉTICOS 2010)

Using this type of equipment, readings and control of physical parameters - such as temperature, fluid flow, humidity, and others - are expected, in order to create an autonomous system which has the ability to both gather data for EWEN - ENERGY's database and send commands to the corresponding equipment whenever necessary.

During the development of the project, another proposal arose to create a pulse counter capable of configuration via a web page where certain parameters - such as number of active channels, and of the network to where the collected data should be sent - are defined. Therefore, it shall be a new equipment which, even distinct from the initial objective, shows promising abilities to be applied in an industrial context.

Key words: Load shedding, energy audit, energy efficiency, parameterization and Analog to Digital Arduino.

Introdução

A proposta apresentada pela EWEN – Energy consistiu no desenvolvimento de um equipamento de aquisição e tratamento de dados com a capacidade de exercer uma resposta em função dos mesmos. Na área energética uma ação deste tipo tem o nome de deslastre de cargas que está definido pelo regulamento de operação das redes elétricas como a interrupção da alimentação de alguns consumos de energia elétrica com o objetivo de preservar o funcionamento do sistema elétrico, a nível local ou nacional, em condições aceitáveis de tensão e frequência. (ENERGÉTICOS, 2010)

A fase inicial de desenvolvimento do projeto consistiu na análise, escolha e a aquisição de *hardware* com as capacidades necessárias para alcançar o objetivo final do projeto. Depois de um estudo de mercado abrangente aos equipamentos com as capacidades necessárias para obtenção de um produto que execute com destreza as funções pretendidas, chegamos à conclusão que a opção mais indicada na relação qualidade /preço seria o Arduíno.

O conceito de Arduíno surgiu em Itália, em 2005, e teve como objetivo a criação de um dispositivo com a capacidade para desenvolver pequenos projetos de forma economicamente mais acessível e de reduzida complexidade de trabalho em comparação com os anteriormente disponíveis no mercado. Este equipamento foi projetado com um conjunto de funcionalidades que possibilitam ao utilizador fácil adaptação e compreensão, apresentando uma programação simples com um conjunto de funções próprias que permitem uma fácil escrita e adaptação às diferentes ideias que se pretendem desenvolver. Outra grande vantagem reside no facto da comunidade que envolve o equipamento ser do tipo *open-source* facilitando largamente a aprendizagem e o rápido desenvolvimento de projetos pelos utilizadores mais inexperientes. Este tipo de abordagem torna também mais rápido e eficiente a construção dos códigos porque grande parte das dificuldades estão já resolvidas e acessíveis a outros utilizadores facilitando e apoiando no desenvolvimento das funções que pretendemos. É um equipamento que funciona como plataforma de computação física pela possibilidade de interação com sensores e controladores onde podem funcionar como mecanismos que permitem a perceção da realidade e a possibilidade de registo ou interpretação dos valores obtidos.

O Arduíno é uma plataforma concebida especialmente para artistas, pensadores e criadores de projetos em geral. O nome Arduíno surgiu usando o nome de um pequeno *pub* em Itália e facilmente este se tornou numa plataforma influenciadora de uma nova geração que pretende desenvolver o mais vasto conjunto de projetos, que pode ir desde projetos universitários até simples projetos desenvolvidos em casa por utilizadores menos experientes apenas por diversão.

O funcionamento do Arduíno é baseado numa placa que tem incorporado um microcontrolador (geralmente do tipo Atmega) com ligação a uma série de portas analógicas e digitais que podem atuar para leitura ou envio de dados interpretando a linguagem de programação própria do Arduíno que é em muito idêntica à das linguagens de programação C ou C++. No fundo todo o equipamento vai funcionar como um pequeno computador contendo microprocessador, memória (*flash* e *RAM*) e periféricos de entradas e de saídas.

As diferentes formas de armazenamento nos Arduínos fixam-se essencialmente em três tipos, a memória *flash* onde são armazenados os programas desenvolvidos, a *SRAM* onde são armazenadas e atualizadas as variáveis (é apagada sempre que existe uma falha na corrente) e a *EEPROM* onde ficam armazenados dados de forma permanente mesmo na ausência de corrente no circuito. As capacidades de todas estas variáveis diferem com modelo de Arduíno escolhido e devem ser tomadas em consideração adaptando-as ao tipo de projeto que se pretende desenvolver.

Uma grande vantagem na utilização de equipamento Arduíno está no alargado conjunto de componentes de hardware que facilitam o desenvolvimento de determinadas aplicações, tal como o uso da placa de ligação Ethernet e de um RTC (“*real time clock”*) como partes essenciais do projeto. A desvantagem na utilização do equipamento Arduíno está na limitação para o desenvolvimento de projetos mais complexos em consequência das suas limitações de memória (limita a extensão do código dos projetos que pretendemos desenvolver) e armazenamento de dados na ausência de memórias externas.

Estabelecidos estes primeiros passos estamos então preparados para o início do desenvolvimento do projeto que passará, numa fase inicial, pela aprendizagem da programação na linguagem do Arduíno que permitirá a construção de rotinas que sejam capazes de executar funções como a leitura de sensores e troca de dados com o servidor, criando uma interface entre a EWEN e o sistema de monitorização.

Os sistemas de monitorização utilizados pelo setor industrial são ferramentas cada vez mais importantes para o controlo eficiente e económico, se necessário, dos equipamentos e recursos que se pretendem analisar em instalações onde a preocupação energética é fundamental. Sendo mais concreto, vamos recorrer ao uso da tecnologia Arduíno com o objetivo de a operacionalizar para uma leitura e controlo de um conjunto de dados recolhidos por sensores de corrente de 4-20mA.

De uma forma mais sucinta o que se pretende com o desenvolvimento deste projeto é a construção de um equipamento com a autonomia para interpretar a realidade e exercer uma resposta previamente definida pelo utilizador de acordo com as suas necessidades. Este tipo de sistemas surge com a necessidade de um sistema capaz de operar e controlar, autonomamente, um conjunto de variáveis eficazmente ultrapassando, por exemplo, anteriores processos manuais de controlo que exigem a disponibilidade de um recurso humano para o fazer. Na construção deste tipo de sistemas deve-se ter em conta o tipo de variáveis que se pretende medir e controlar e a exigência temporal de resposta que é pretendida para cada caso em particular.

Para completar uma montagem final capaz de medir e controlar todos os parâmetros físicos presentes na instalação foi proposta a criação de um equipamento capaz de proceder à leitura e envio periódico dos dados recolhidos dos diferentes sensores que se pretendem parametrizar numa instalação. Este equipamento será projetado para a leitura de contadores de energia por impulsos e o posterior envio da informação para servidores que terão como função o trabalho dos dados com o objetivo de formar um conjunto de informação capaz de responder às necessidades pretendidas pelo cliente.

Assim no final teremos dois equipamentos distintos em que um deles efetuará processos de deslastre para o controlo de equipamentos na instalação e o complementar que terá como função a leitura dos parâmetros físicos enviados por impulsos até ao Arduíno e posteriormente reencaminhados periodicamente, em intervalos personalizados pelo utilizador, para um servidor.

Esquematização das abordagens de resolução

Inicialmente foi necessário pensar conceptualmente em todo o problema para permitir a possibilidade de esquematização da sequência de passos a percorrer para permitir uma evolução organizada do projeto com vista a obtenção de um produto final com capacidade para ser implementado em contexto industrial.

Numa instalação típica existe um alargado número de sensores sob os quais é necessário um controlo adequado como forma de garantir o bom funcionamento de todo o esquema industrial; o objetivo será ter um controlo e parametrização de um conjunto de equipamentos. Sendo que é necessário ter em conta diferentes tipos de abordagens para que seja possível desenvolver um sistema com capacidade para efetuar deslastre, será importante equacionar um alargado número de condições com objetivo de prever os cenários reais com que nos iremos deparar. À partida numa instalação teremos que responder a condições como: os tipos de parâmetros físicos que pretendemos que sejam medidos, as distâncias entre os diferentes sensores a controlar, de que forma será estabelecida a comunicação entre eles, a necessidade ou não de mais do que um equipamento para o armazenamento e envio das leituras de dados, entre outras condições. Tendo em conta as diferentes variáveis que se pretende medir foram consideradas diferentes opções de resolução para o problema.

Primeira Opção

Numa primeira opção consideramos o problema de uma instalação para a qual será necessário o controlo e ou parametrização de um elevado número de sensores onde a distância faz com que não seja viável o estabelecer de uma ligação física entre eles e um concentrador comum para o envio dos dados para o exterior recorrendo a cabos, que com o aumento da distância vão levar a um aumento considerável no custo final e na complexidade de instalação. Na ocorrência de casos com estas condições, a solução que se pensou consiste na ligação de sensores a um conjunto de concentradores individuais que estarão ligados a um concentrador geral por ligação MODBUS, que é um tipo de ligação presente já na maioria das indústrias. O esquema geral consiste assim na recolha de dados através dos sensores por parte de equipamentos Arduínos,

previamente programados, sendo posteriormente enviados por MODBUS para um concentrador principal que será o responsável pelo envio de todos os dados para o servidor, neste caso, da EWEN.

A vantagem desta abordagem é que apenas um dos Arduínos terá a função de envio dos dados sendo por isso necessário abrir portas Ethernet de saída e entrada de dados para o exterior da empresa em apenas um dos dispositivos. Cada um dos dispositivos terá uma identificação específica para que posteriormente a recolha dos dados possa ser efetuada de forma autónoma pela base de dados da EWEN que vai funcionar como sistema de monotorização ao qual os respetivos clientes terão acesso para análise dos valores de operação lidos pelos sensores nas suas instalações. Outra vantagem importante está na comunicação de todos os dados para um concentrador comum que terá como responsabilidades o armazenamento e envio dos dados para um servidor para que posteriormente possam ser analizados. O esquema ilustrativo deste tipo de abordagem pode ser visualizado na Figura 1.

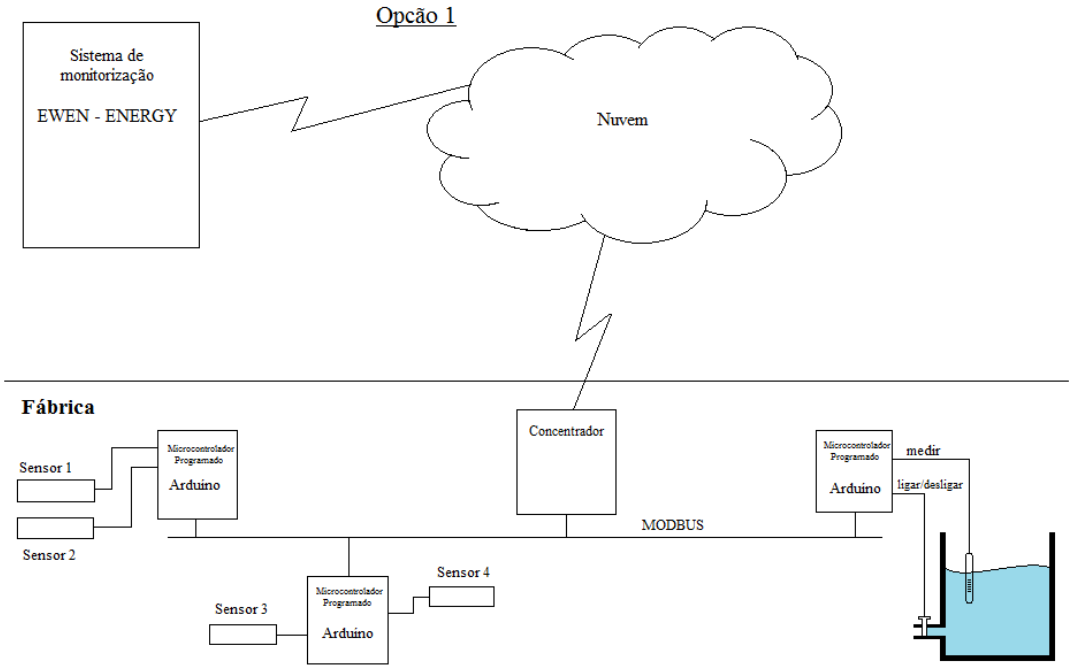

Figura 1: Imagem descritiva da opção 1

Segunda Opção

A segunda opção, ilustrada na Figura 2, é direcionada para casos de clientes com necessidade de parametrização de um número mais reduzido de sensores. Neste tipo de casos em particular a recolha dos dados poderá ser efetuada por um reduzido número de Arduínos que serão também os responsáveis pelo envio direto dos dados para o servidor da EWEN, o que vai descartar a exigência de comunicação MODBUS entre eles.

A desvantagem no uso deste tipo de abordagem está na necessidade de abertura de mais do que uma porta de Ethernet para comunicação e troca de dados com o exterior pelo cliente, o que pode ser uma dificuldade na aceitação do projeto. Como vantagem neste tipo de aplicação temos a simplicidade na instalação dos sistemas, com a necessidade apenas de ligação dos sensores ao Arduíno sendo a comunicação dos dados responsabilidade individual de cada um deles. Esta opção exige apenas a ligação entre os sensores e o Arduíno permitindo assim uma redução na possibilidade de ocorrência de falhas e a consequente perda de dados.

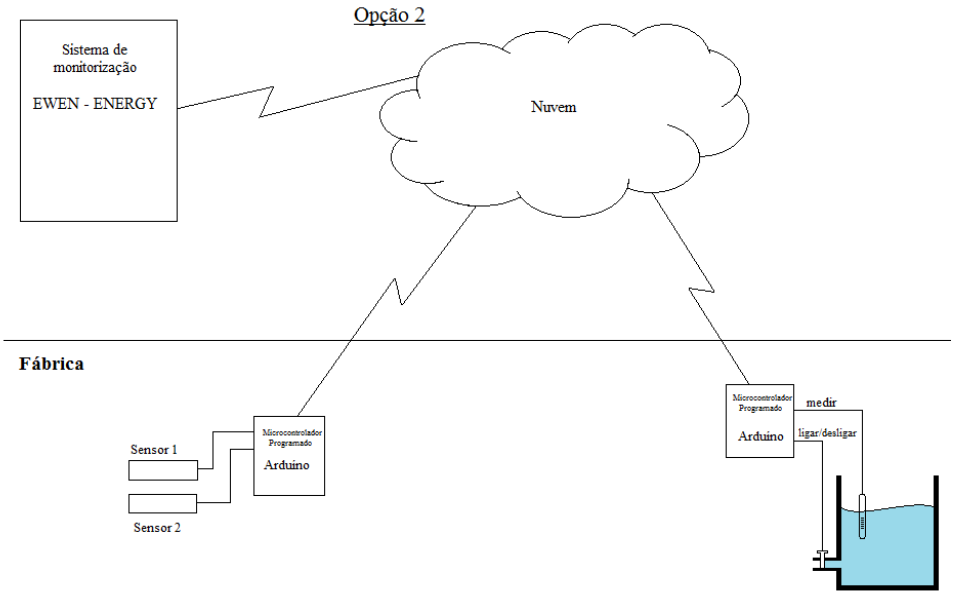

Figura 2:Imagem descritiva da opção 2

Terceira Opção

A terceira opção está direcionada para os casos em que existe a possibilidade de comunicação de Ethernet para todos os equipamentos instalados. Neste tipo situação a melhor solução é a ligação por internet dos Arduínos a um concentrador que irá ser responsável pela comunicação dos dados com o servidor exterior. Este tipo de abordagem pode também permitir que os equipamentos tenham a possibilidade de comunicarem mais facilmente entre si, uma vez que a comunicação Ethernet dos Arduínos é mais eficaz que a comunicação MODBUS, garantindo assim a comunicação e troca de dados mais simplificada e com mais possibilidades na leitura de diferentes tipos de parâmetros. Este tipo de abordagem pode ser observado na Figura 3.

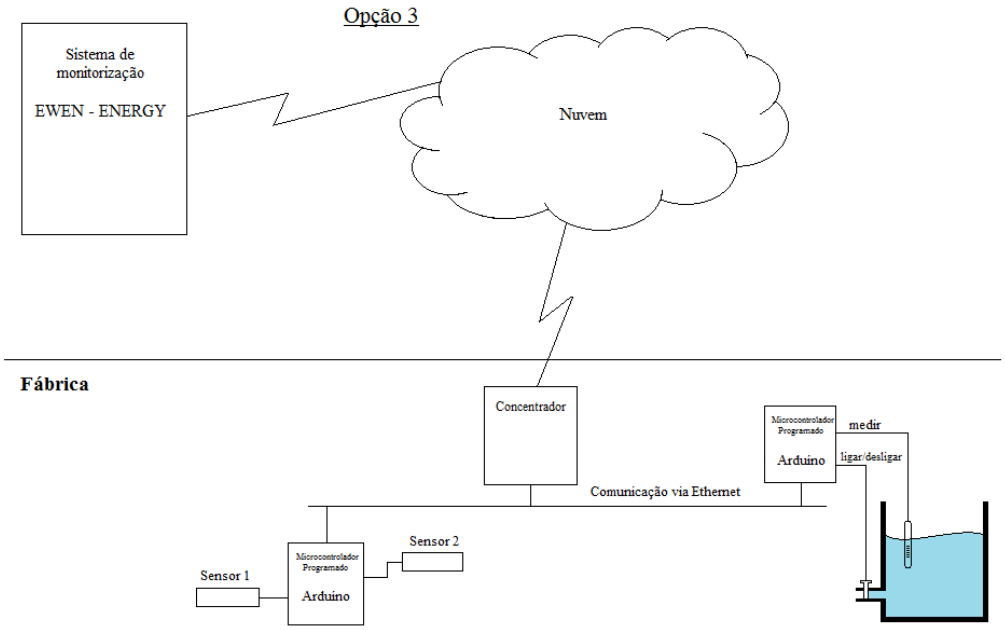

Figura 3: Imagem descritiva da opção 3

Estado da arte

Existe um conjunto relativamente alargado de instrumentos com a capacidade para leitura e envio de dados correspondentes a parâmetros físicos analisados por sensores que poderão ser conectados a um conjunto de entradas disponíveis no equipamento. Nos últimos meses aumentou o número de microcontroladores disponíveis no mercado,

sendo que grande parte desse facto deve-se ao grande aumento de clones do Arduíno que têm vindo a aparecer nos últimos tempos. (Make:, 2014)

Vou apresentar de uma forma geral os instrumentos disponíveis, de momento, no mercado. Um dos microcontroladores disponíveis é o BeagleBone Black, representado na Figura 4, que é o novo instrumento da família BeagleBone e trata-se de um equipamento com um custo de 36€, que é bastante mais reduzido que o apresentado pelo anterior modelo BeagleBone com um custo 70€. A redução no custo deve-se ao facto deste novo modelo utilizar um processador de baixo custo da *Texas Instruments*. Este equipamento suporta o sistema operativo Linux, que inicia ao final de 10 segundos, e necessita de 5 minutos para estabelecer a comunicação USB que permite o desenvolvimento do software para o mesmo. Uma nova característica deste é a possibilidade de passagem do sistema operativo para a memória flash libertando espaço no SD que pode ser usado para outros propósitos. Com estas características o BeagleBone Black torna-se um competidor direto do Raspberry Pi com a vantagem de ser mais flexível na possibilidade de ligação com *hardware* externo.

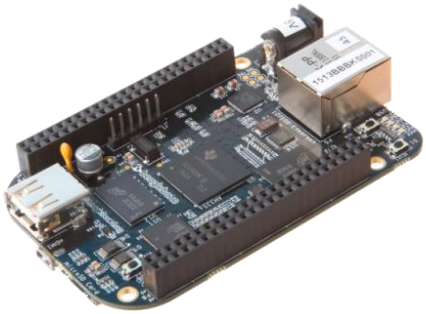

Figura 4: BeagleBone Black (Make:, 2014)

Recentemente lançado no mercado temos também o UDOO que trata de um equipamento que possui uma placa baseada no sistema operativo Linux (tal como o *Rapberry Pi*) e uma segunda parte que contém um processador que tenta imitar o Arduino *Due* no número de entradas e saídas analógicas e digitais. Trata-se assim de um pequeno computador que corre em Linux ou em *Android*, com um CPU dual ou *quad*-core, contém também gráficos integrados e 54 portas digitais I/O, portas analógicas, *Ethernet*, *on-board* WiFi, HDMI, USB, SATA e áudio analógico. Um modelo com versão de *quad core* tem um preço a retalho de 102€. Este instrumento está apresentado na Figura 5.

Figura 5: UDOO (UDOO, 2014)

Desenvolvido pela empresa Australiana Pozible temos disponível o equipamento com o nome The Goldilocks (Figura 6). Este instrumento trata-se de um clone do Arduíno, que substitui o microcontrolador ATmega328P do UNO ou o ATmega2560 do Mega pelo ATmega1284P que contém os mesmos fatores de forma do UNO, com a vantagem de possuir uma maior capacidade de memória SRAM, qualquer coisa como oito vezes mais que o UNO e duas vezes mais que o Mega. É um bom equipamento substituto do Arduíno UNO para utilizadores que lutam contra limitações de memória SRAM, tendo no entanto um custo de 36€.

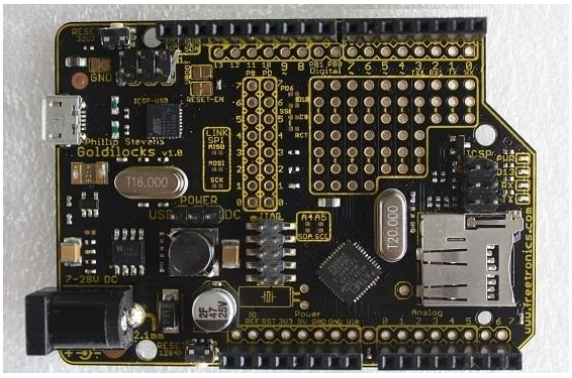

Figura 6: The Goldilocks (Feilipu, 2014)

Outro equipamento disponível é o DigiX (Figura 7) que se trata de um instrumento que tenta ser um pouco de tudo e para todas as aplicações. É baseado no Arduíno DUE que contém uma placa WiFi de baixa energia, um total de 99 pins I/O, um *real time clock*, 4xUARTs (portas serial), 2× I2C (*Two-wire-interface*), SPI (*Serial*

Peripheral Interface), CAN Bus, 2× DAC, JTAG, e DMA. Mesmo com este elevado número de capacidades o seu preço a retalho são uns impressionantes 47€.

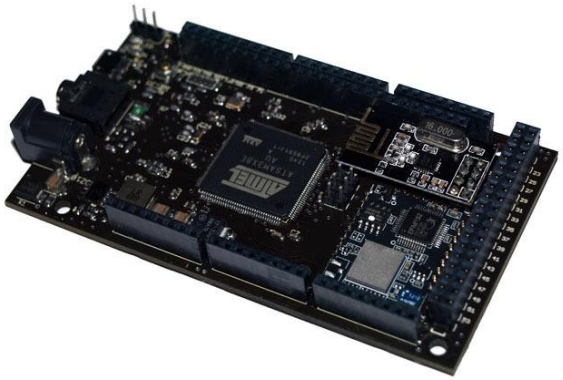

Figura 7: DigiX (kickstarter, 2014)

Por fim podemos ter em consideração o pcDuino que se trata de uma placa que contém o sistema operativo Linux incorporado e está fisicamente apresentado na Figura 8. Tem a característica interessante de possuir um conjunto de pins compatíveis com Arduíno o que possibilita a hipótese de trabalho com qualquer um dos componentes extra disponíveis para o Arduíno. Este tem também a hipótese de programação interagindo diretamente com a placa tal como é feito com o Arduíno, e aparenta ser de fácil utilização. O custo deste tipo de equipamento no mercado é de 48€.

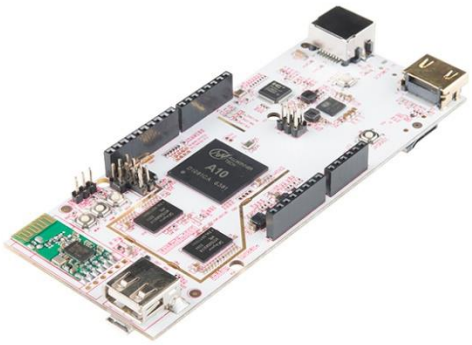

Figura 8: pcDuino (SparkFun Electronics [US], 2014)

Introdução ao Arduíno

Um dos principais fatores do sucesso da plataforma Arduíno consiste na quantidade reduzida de tempo que é necessário despender para que um utilizador inexperiente consiga escrever o seu primeiro código e interagir com a placa. O *software* necessário para a comunicação com a placa está disponível na página oficial do Arduíno livremente, e funciona bem nos sistemas operativos *Mac*, *Linux*, *Windows*.

O Arduíno é uma placa de interface que contém um microcontrolador no seu sistema, e podemos assim observar o conjunto como um pequeno computador num chip.

O microcontrolador utilizado pelo Arduíno é da origem de uma empresa chamada *Atmel* e o chip é conhecido como AVR. Considerando as capacidades dos computadores existentes no mercado pode ser visto como um microcomputador lento, isto porque corre a uma frequência de apenas 16 Mhz com um núcleo de 8-bits e apresenta também capacidades limitadas de memória com 32 Kbits reservados para armazenamento e 2 Kbits de memória volátil.

Este tipo de estrutura baseada na junção de um microcontrolador de baixo custo e uma saída USB-to-serial não foi revolucionária no mercado, visto existirem já equipamentos com o mesmo tipo de formato, no entanto o Arduíno é o único cujo *software* de comunicação computador-Arduíno é livremente distribuído pelos utilizadores ao contrário dos restantes como o OOPIC, Basic ATOM, BASIC-X24 e o PICAXE. O baixo custo com que são apresentados no mercado, cerca de metade dos equipamentos com capacidades idênticas, torna-os uma ferramenta muito apetecível para os desenvolvedores de projetos de todo o mundo. Igualmente importante no ambiente de Arduíno é o seu programa de processamento onde é possível escrever, editar, compilar e fazer o "*upload*" para o Arduíno de um código que irá transmitir instruções à placa microcontroladora para que realize as ações pretendidas pelo utilizador. Baseados na ideia de que os utilizadores do Arduíno apenas pretendiam a criação de projetos divertidos, sem a preocupação de toda a arquitetura por de trás do microcontrolador os criadores desenvolveram esse *software* que contorna todo esse tipo de especificações tornando a programação mais simples e de elevada facilidade de compreensão. Esta plataforma é tão versátil que a sua utilização para a deteção de erros na escrita do código não necessita de uma ligação física a uma placa de Arduíno. (Ladyada, 2014)

A plataforma de Arduíno é muito útil para a criação de projetos usando microcontroladores, mas esse motivo não é suficientemente forte para a sua larga divulgação e popularidade da plataforma por parte dos utilizadores. Outra grande vantagem de utilização reside na facilidade com que se pode adquirir *hardware* do tipo Arduíno bem como todo um outro tipo de acessórios compatíveis com o mesmo que possibilitam a realização e concretização de projetos até aos limites que a imaginação criativa nos permita alcançar.

Sabendo da importância da plataforma física que assegura o Arduíno e não desvalorizando a mesma, podemos assegurar que uma das mais importantes características deste equipamento está na grande comunidade que este atualmente envolve. Este tipo de suporte permite que facilmente possam ver resolvidas e explicadas questões que surjam durante o desenvolvimento de projetos. Esta comunidade de criadores contribuiu para o Arduíno com um desenvolvimento de novos códigos, bibliotecas e mostrando todo o tipo de novos projetos que se têm vindo a desenvolver.

Podemos observar um exemplo de placa Arduíno na Figura 9, onde estão todos os componentes principais para o bom funcionamento do equipamento.

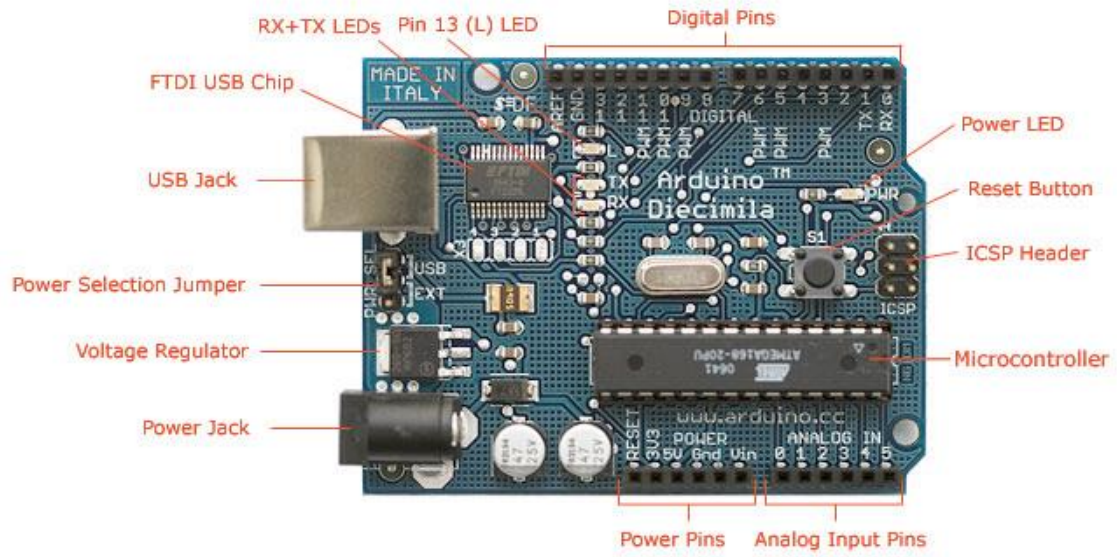

Figura 9: Descrição da composição de uma placa Arduino (Schuman, 2014)

Neste momento existe já no mercado um conjunto alargado de diferentes modelos de Arduínos com as mais diferentes características, com vista a possibilitar e alargar as possibilidades de aplicações do mesmo nos mais variados projetos. É possível desta

forma que para um criador a plataforma de Arduíno se transforme numa base de teste para o lançamento de projetos que podem posteriormente ser adaptados em modelos mais desenvolvidos e aplicados no mercado. Na Tabela 1 apresenta-se uma lista do vasto conjunto de equipamentos disponíveis no mercado pela Arduíno.

Tabela 1:Tabela de comparação dos modelos de Arduíno (Arduino, 2014)

Nome	Velocidade CPU (Mhz)	Analog In/Out	Digital IO/PW	EEPROM [KB]	SRAM [KB]	FLASH [KB]
Uno	16	6/0	14/6	1	2	32
Due	84	12/2	54/12	-	96	512
Leonardo	16	12/0	20/7	1	2.5	32
Mega 2560	16	16/0	54/15	4	8	256
Mega ADK	16	16/0	54/15	4	8	256
Micro	16	12/0	20/7	1	2.5	32
Mini	16	8/0	14/6	1	2	32
Nano	16	8/0	14/6	0.512/1	1/2	16/32
Ethernet	16	6/0	14/4	1	2	32
Esplora	16	-	-	1	2.5	32
ArduinoBT	16	6/0	14/6	1	2	32
Fio	8	8/0	14/6	1	2	32
Pro(168)	8	6/0	14/6	0.512	1	16
Pro(328)	16	6/0	14/6	1	2	32
Pro Mini	8	6/0	14/6	0.512	1	16
LilyPad	8	6/0	14/6	0.512	1	16
LilyPad USB	8	4/0	9/4	1	2.5	32
LilyPad Simple	8	4/0	9/4	1	2	32
LilyPad SimpleSnap	8	4/0	9/4	1	2	32
Duemilanove	16	6/0	14/6	1	2	32

Características do Duemilanove

No desenvolvimento deste projeto foi utilizado o modelo Duemilanove da Arduino que apresenta dois modelos diferentes onde um tem como base o microcontrolador ATmega168 e o outro com mais capacidades que tem por base um microcontrolador ATmega328. Este equipamento tem embutido na sua placa 14 portas digitais que podem funcionar como entradas ou saídas em que seis podem ainda funcionar como saídas de modulação de impulsos, conseguindo assim uma resolução de 1024 ao contrário do funcionamento binário normal de 0 ou 1. Contém também 6 entradas analógicas, um oscilador de cristal de 16 MHz, entrada de alimentação, conetor ICSP e botão de reiniciar.

Na

Tabela 2 podem ser analisadas as características típicas deste modelo de Arduíno.

Tabela 2:Ficha de descrição do Arduíno Duemilanove (utilizado no projeto)

Arduino Duemilanove	
Microcontrolador	ATmega168 / ATmega328
Portas digitais	14 (6 com PWM – modelação de impulsos)
Portas analógicas	6
Tensão de operação	5 V
Tensão de alimentação (recomendada)	7 - 12 V
Limites de tensão de alimentação	6 – 20 V
Corrente contínua por pino	40 mA
Memória Flash	32 KB
SRAM	2 KB
EEPROM	1 KB

Tipo de Linguagem do Arduíno

A linguagem principal usada pelo Arduíno é a linguagem de programação C, linguagem desenvolvida no início da década de 70 para uso com o sistema operativo UNIX. Os

criadores da plataforma Arduíno inteligentemente desenvolveram um conjunto de bibliotecas básicas que proporcionam ao Arduíno múltiplas funções que tornam a programação inicial muito simples, mesmo para os que se estão a iniciar no mundo da programação. É importante mencionar que as bibliotecas são na realidade escritas em linguagem C++ que é uma derivada da linguagem C, sendo que na linguagem Arduíno as funções permanecem fundamentalmente iguais às praticadas em C, no que respeita à sintaxe, estrutura, operadores e propriedades básicas. Existem algumas funções como *pinMode()*, *digitalWrite()*, *digitalRead()*, *analogRead()*, *delay()* que são exclusivas do Arduíno e de elevada utilidade na criação de qualquer tipo de código que envolva a interação do Arduíno com as suas portas analógicas e digitais.

Para iniciação na linguagem Arduíno o mais importante a perceber está na estrutura geral de configuração que todos os projetos terão que respeitar. Esta divide-se essencialmente em duas partes: "*setup"* e "*loop"*.

O s*etup* é apenas percorrido uma única vez no código pelo que será neste que iremos definir todas as funções gerais do código que apenas necessitem de ser estabelecidas uma vez.

O *loop* é uma função que será percorrida em ciclo durante o restante tempo de operação do equipamento e será portanto nesta que serão definidas todas as funcionalidades que se pretendam atribuir ao projeto e que necessitem de atualização permanente para garantir o bom funcionamento do mesmo.

Para além destas duas funções gerais e obrigatórias é sempre possível a criação de funções à parte mas que para serem percorridas terão de ser mencionadas numa das funções anteriores.

Para uma melhor perceção do tipo de funções básicas que devem ser apresentadas nesta linguagem é possível observar o exemplo de código na Figura 10 e na Figura 11, onde no primeiro caso temos a leitura do valor registado pela entrada analógica zero e o imprimir do mesmo na porta serial para que possa ser observado qual o seu valor. A porta serial representa a janela onde podem ser enviados e recebidos valores do Arduíno.

```
void setup() {
  Serial.begin(9600);
}

void loop() {

  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
  delay(1);
}
```

Figura 10: Ilustração da leitura de uma entrada analógica

Neste caso da segunda figura temos o exemplo típico de interação e troca de informação entre as entradas/saídas digitais para obtenção de um resultado final em que neste exemplo concreto da Figura 10 temos um botão, associado à porta digital 2, que ativa o led da placa Arduíno que está ligado à entrada digital 13.

```
const int buttonPin = 2;
const int ledPin =  13;


int buttonState = 0;

void setup() {

  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop(){
:
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
  }
  else {
    digitalWrite(ledPin, LOW);
  }
}
```

Figura 11: Exemplo de interação entre entradas e saídas de portas digitais

Para os utilizadores que pretendem uma utilização mais profunda e mais perto da linguagem do computador, o Arduíno é totalmente compatível e extensível a programação direta usando linguagem mais próxima da linguagem mais básica com o auxílio da biblioteca *avr-libc* e um compilador GCC (*GNU Compiler Collection*) onde

ambos são especificados para microcontroladores padrão Atmel AVR de 8-bits. Este tipo de programação pode ser observado na Figura 11, e por norma ocupa muito menos espaço na memória disponível para o código do microcontrolador que o ocupado pela linguagem própria do Arduíno**.**

```
#include <avr/io.h>
#include <util/delay.h>

int main(void) {
  while (1) {
    PORTB = 0x20;
    _delay_ms(1000);
    PORTB = 0x00;
    _delay_ms(1000);
  }
  return 1;
}
```

Figura 12: Programação com avr-libc

O código anterior apresenta as mesmas funções que o código da Figura 11, no entanto apenas ocupa um quinto da memória.

A desvantagem é que este tipo de linguagem é muito menos intuitiva que a linguagem própria Arduíno. Por exemplo na linguagem de Arduíno para ligar o LED que está conectado ao pin 13 um simples *digitalWrite()* é suficiente, enquanto usando a livraria *avr-libc* o que é necessário fazer é enviar o valor hexadecimal 0X20 para a PORTB. Assim o tipo de linguagem Arduíno C tem uma grande vantagem na sua simplicidade de utilização e leitura de código por parte dos utilizadores, sendo portanto muito importante no processo de utilização por parte de programadores principiantes neste tipo de abordagem como se verificou ser o meu caso aquando do início deste projeto.

A limitação deste método de escrita de informação para o Arduíno prende-se com o reduzido número de funcionalidades que se podem desenvolver de forma simplificada, pelo que para a criação de funcionalidades mais complexas e elaboradas não é viável o uso deste tipo de linguagem.

Este tipo de abordagem pode-se tornar adequada para utilizadores que pretendam a compreensão de todo o tipo de processos de mais baixo nível que estão na base de construção das funcionalidades mais complexas. Pode ser muito interessante o uso deste método por parte de todos os utilizadores para um sentido mais educativo com o objetivo de funcionar mais tarde como ferramenta para resolução de problemas que eventualmente podem e irão acontecer.

Descrição formal do código Arduíno

Uma característica importante na estrutura do código de Arduíno é a execução sem interrupções de todas as funções presentes no código, isto é, o código usa a ordem pelo que são introduzidas as funções no *loop()*, avançando apenas para a função seguinte quando executada a anterior. A desvantagem deste tipo de estrutura é o fato de quando não é executada uma determinada função ou ciclo o restante código não é acedido e passado um tempo limite de um acontecimento deste tipo é automaticamente reiniciado o Arduíno voltando de novo ao *Setup()*, este reiniciar é uma capacidade de proteção do para quando a funções excedem as capacidades de memória do mesmo. Quando o código fica bloqueado por um ciclo ou a aguardar que uma dada função seja verificada então neste caso ele permanece nesta sem que seja reiniciado todo o código.

Variáveis

As variáveis são funções essenciais para o desenvolvimento de qualquer código que têm como principal função o armazenamento e transporte de informação entre funções, ou dentro da mesma função, auxiliando na criação de iterações entre diferentes processos criados no desenvolver de projetos.

Para uma abordagem inicial descrever-se-á como declarar uma variável. Nesta linguagem, em semelhança com muitas outras para declarar uma variável é necessário indicar o tipo e o nome da variável como por exemplo:

```
int x;
```

Este formato indica uma variável do tipo inteiro com o nome de *x.* O tipo de dados que podem ser guardados são indicados pelo tipo definido para a mesma.

Na declaração de uma variável é possível atribuir à mesma um valor inicial. Como se pode ver no exemplo a seguir, em que temos uma variável do tipo inteiro com o nome *count* e um valor inicial igual a 10.

```
int count=10;
```

No que diz respeito à definição dos nomes das variáveis, podem ser atribuídos todo o tipo de combinações possíveis à exceção do espaço, @ ou &. Na construção concreta de nomes não é também possível atribuir nomes de funções às mesmas, como por

exemplo, delay(), pinMode(), e todos os nomes correspondentes a funções predefinidas no código. Quando é atribuído o nome a uma variável e este nunca pode coincidir com nomes de funções próprias caraterísticas da linguagem. As variáveis podem ser definidas como locais ou globais, sendo que as variáveis locais, por definição da linguagem, ocupam menos espaço na memória RAM e para definir uma variável deste tipo apenas terá que ser definida no interior de uma função, não permitindo a possibilidade de ser usada noutras funções ao contrário das globais que podem ser usadas em todas as partes do código.

Tipo de dados

Existem dois tipos de dados diferentes que podem ser definidos: *signed* ou *unsigned*. No caso das variáveis definidas como *signed* estas podem adquirir valores entre -32,768 e 32,767, enquanto uma variável definida como *unsigned* apenas pode ser definida como valor positivo e tem um alcance de 0 até 65,535. Os números inteiros no código de Arduíno ocupam em geral o espaço de 16 bits que são o equivalente a 2 bytes da memória total de 32 Kbytes.

Por exemplo se considerarmos uma variável x *signed* e lhe forem atribuídos os valores do seguinte exemplo:

```
int x = 32767;
x = x + 1;
```

Neste caso em particular o valor da variável x vai passar a ser -32,768 e o ciclo vai-se assim repetindo sempre que atinge um valor que ultrapasse os limites.

Considerando agora o caso de valores com vírgulas flutuante (*float*) estes ocupam o espaço total de 32 bits e tal como os valores inteiros podem ser positivos, negativos ou zero. Os valores decimais são declarados da forma apresentada no exemplo seguinte:

```
float num=2.345;
```

No entanto este tipo de variáveis nem sempre apresenta os melhores resultados na precisão e sendo que se estende a apenas 6 ou 7 casas decimais será importante um cuidado e atenção para casos em que a exigência de um maior rigor de precisão signifique um fator crítico para o desenvolver do projeto pretendido pelo utilizador.

As variáveis podem ainda ser definidas como sendo do tipo Byte. As características deste tipo são o tamanho limite de 8 bits o que significa portanto que podem tomar valores de 0 a 255. Dada a sua reduzida dimensão são variáveis que necessitam de muito pouco espaço na memória o que faz delas um excelente método de recurso para redução da memória utilizada, onde estas facilmente substituem variáveis destinadas para o armazenamento de valores pequenos.

Na Tabela 3 é possível observar uma descrição sucinta das dimensões de cada um dos tipos de variáveis existentes no Arduíno.

Tabela 3:Descrição geral das dimensões das diferentes variáveis

Nome	Tamanho (bit)	Alcance
boolean	1	true or false
byte	8	0 a 255
char	8	-128 a 127
int	16	-32,768 a 32,767
long	32	-2,147,483,648 a 2.147.483.647
float	32	-3.4028235E+38 a 3.4028235E+38

Avançando para o tipo de variáveis positivas recorremos ao uso da palavra *unsigned* na sua declaração para garantir que estas apenas possam assumir valores positivos alterando assim os seus limites, tal como podemos verificar na Tabela 4. Este tipo de variáveis torna-se importante principalmente na implementação de um sistema de contagem salvaguardando ao máximo o alcance de cada contador que se pretende implementar sem que aconteça *overflow* e sejam perdidos os valores da contagem.

Tabela 4: Descrição do tamanho das variáveis *unsigned*

Nome	Tamanho (bit)	Alcance
unsigned char	8	0 a 255
unsigned int	16	0 a 65,535
unsigned long	32	0 a 4,294,967,295

Existem ainda um conjunto de funções de estado que estão predefinidas no código Arduíno sendo estas o *true* (verdadeiro) ou *false* (falso) bem como as variáveis usadas para a operação com os pins que são *INPUT*, *OUTPUT*, *HIGH* e *LOW.* É de notar que as duas primeiras são apenas reconhecidas quando escritas em minúsculas enquanto

as quatro últimas em maiúsculas. Sendo muito importante realçar que estas quatro últimas são extremamente uteis para o desenvolvimento de projetos que recorram a trocas de informação através das entradas/saídas digitais e analógicas.

Arrays

Uma variável pode ainda ser transformada numa lista de múltiplas variáveis onde os seus elementos são indicados pelo número do índice correspondente à posição em que se encontra. Este tipo de variável é muito útil para a atribuição de listas ou conjuntos de valores recorrendo assim apenas a uma variável responsável pelo armazenamento de todos em diferentes índices. A declaração de um *array* é semelhante à de uma variável como podemos ver no exemplo seguinte:

```
int list[5];
```

Ou então também pode ser definido com a atribuição de valores aos seus campos como podemos ver no exemplo seguinte:

```
int list[]={0,2,5,1,0};
```

Neste último caso temos a inicialização de um *array* com valores já previamente atribuídos a cada um dos elementos. É importante mencionar que nos índices referentes aos elementos presentes no *array* o valor zero corresponde sempre ao primeiro elemento.

Para além da construção de *arrays* de números inteiros podemos também construir *arrays* de caracteres permitindo a possibilidade de guardar dados de texto como *strings*. Uma forma de construção pode ser a seguinte:

```
char text[]= "FCUP/EWEN";
```

Recorrendo a este tipo de construção podemos aceder a qualquer um dos caracteres usando o número do índice correspondente o que é uma mais-valia para projetos em que a avaliação de variáveis do tipo texto percorrendo todos os elementos seja necessária. Pode por exemplo ser usada em casos que necessitem de encontrar um determinado carater específico contido numa *string* sendo esta uma forma de aceder a cada um dos elementos presentes na variável de texto. O código permite ainda a construção de *arrays* multidimensionais que representam *arrays* dentro de *arrays* o que permite a possibilidade de aplicação a todo um conjunto de construções de

complexidade mais elevada. Usando esta funcionalidade facilmente conseguimos a construção de matrizes uteis para módulos matemáticos ou gráficos que se pretendam implementar.

Operadores aritméticos

O código, tal como a maioria das linguagens de programação, possui um conjunto de operadores que permitem as iterações aritméticas básicas entre os carateres do tipo numérico. Podemos assim definir e executar todo o tipo equações e enviar para o microprocessador processar e enviar o resultado.

Os operadores aritméticos presentes no código C de Arduíno são a soma (+), a subtração (-), a multiplicação (*) e a divisão (/). Para uma pequena introdução às operações podemos observar o exemplo seguinte da soma do valor de uma variável com um outro valor não declarado em variáveis.

```
int valor=12;
valor = valor - 10;
```

No exemplo anterior o que podemos constatar é a definição de uma variável à qual é atribuído o valor 12 e posteriormente definida como sendo igual ao valor inicial menos 10. A variável com o nome *valor* adquire o valor final de 2 a partir da segunda atribuição.

Considerando agora outro caso, interessante do ponto de vista de definições de tipos de variável, em que a operação entre dois números de um mesmo tipo tem como resultado um valor de um outro tipo, como no exemplo seguinte:

```
int valor=5;
valor = valor/2;
```

No exemplo apresentado o que temos é a divisão de dois valores inteiros que têm como resultado um valor decimal, mas como a variável valor está definida como tipo inteiro então o resultado desta operação será o valor truncado ao valor inteiro resultante da operação, ou seja, o valor final é 2.

Tal como em aritmética básica existe uma ordem segundo a qual as diferentes operações se irão realizar e no código de Arduíno isso não será exceção, pelo que na

Tabela 5 está apresentada de forma clara e sucinta a ordem com que são realizadas as operações aritméticas mais importantes para a compreensão do código do Arduíno.

Tabela 5: Descrição dos operadores matemáticos característicos da linguagem

Símbolo	Descrição	Exemplos
()	Parenteses	A * (B – C)
++ ; --	Aumento, diminuição	A++ ,B--
* ; / ; %	Multiplicação, divisão e módulo	A*2, B/5
+ ; -	Adição e subtração	A+B
< ; > ; >= ; >=	Comparações de maior e menor	if (A>=30)
== ; !=	Igual e diferente	if (A==LOW)
&&	E lógico	if (A==HIGH && A < 30)
||	Ou lógico	if (A==HIGH || A < 30)
+= ; -= ; *= ; /=	Atribuição e atribuição composta	A+=B

Ciclos

- FOR

O ciclo *for* é uma declaração que permite ao Arduíno executar uma determinada ação repetidamente o número de vezes que pretendermos. Por exemplo quando pretendemos que um array seja percorrido pelos seus índices, em ciclo, numa execução rápida e simples que permite trabalhar um conjunto alargado de valores. Para definir um ciclo *for* é necessário definir a variável que vai ser iterada, o limite da iteração e o tipo de incremento que se pretende efetuar. No exemplo seguinte podemos observar a constituição básica deste tipo de ciclo.

```
for (int i=0; i<10 ; i++){
 //Colocar a afirmação
}
```

- WHILE

O ciclo *while* permite que uma determinada ação seja executada até que uma determinada condição se verifique, não necessitando de uma declaração prévia do número de vezes que irá ser executada. Um exemplo de construção de um ciclo *while* básico poderá ser o seguinte:

```
int a = 0;
while (a<5){
  digitalWrite(pin[0],HIGH);
  delay(1000);
  a++;
}
```

Neste ciclo o que acontece é que o pin vai enviar o sinal um enquanto a condição não se verificar, como temos um *delay* de 1000*ms*, o que acontece é que o pin vai ficar a enviar o valor 1 (HIGH) durante 5 segundos que irá ser o tempo necessário até a condição deixar de se verificar.

A vantagem do ciclo *while* perante o *for* é que este permite a declaração de um ciclo que atue repetidamente sem que seja necessário o conhecimento prévio do número de iterações a efetuar até que se verifique a condição pretendida. A desvantagem deste tipo de ciclo é a interrupção do seguimento sequencial do código que permanece no ciclo até que determinada condição se verifique, transformando assim todo o código dependente de uma verificação. Para casos particulares esta desvantagem pode ser propositadamente usada pelo utilizador para impor determinadas caraterísticas de imposição de condições para o desenrolar dos restantes processos.

Funções de verificação

- DO

O ciclo *do* é apenas uma extensão do ciclo *while* mas este executa as linhas de código e apenas depois é que efetua a verificação da condição para poder continuar o código. Um exemplo simples deste tipo de código é o seguinte:

```
int a=0;
do{
  Serial.println(a);
  a++;
}while (a<5);
```

Neste exemplo o valor da variável *a* é impresso na porta serial sempre que a condição é verificada, ou seja, vamos ver uma contagem de 0 a 5 acontecer na porta serial.

- SWITCH

O *switch,* do meu ponto de vista, trata-se apenas de uma forma mais elegante de apresentar as condições *if*, permitindo um controlo mais eficaz quando estamos perante um tipo de estrutura de código mais complexa. Usando este tipo de funções podemos tornar o nosso código mais intuitivo e de menor complexidade de leitura, constituindo uma grande vantagem, por exemplo, para a procura de erros em códigos extensos. Um exemplo de código usando este método é o seguinte:

```
switch (digitalRead(5)>){
  case 0:
    digitalWrite(3 , LOW);
    delay(2000);
  case 1:
    digitalWrite(3 , HIGH);
    delay(2000);
}
```

O que acontece neste exemplo é o envio de um sinal como sendo 1 para a porta digital 3 sempre que na porta digital 5 seja detetado sinal com o valor 1. Neste exemplo, no interior do Setup, a porta digital 3 terá de ser declarada com saída (OUTPUT) e a porta digital 5 como entrada (INPUT).

- BREAK

A declaração do *break* é efetuada no interior de um ciclo e tem como função fazer com que este seja imediatamente interrompido sempre que seja verificada a condição em que este se encontra. Pode por exemplo ser usado quando esperamos que a partir de um determinado valor o ciclo seja interrompido. No exemplo seguinte podemos observar melhor o uso do *break* no interior de um ciclo *for*:

```
for (int i=0; i<10 ; i++){
  if (analogRead(A0)<200) break;
  else{
    digitalWrite(13,HIGH);
  }
}
```

No exemplo anterior o que acontece é que durante a execução do ciclo sempre que a entrada analógica enviar um valor menor que 200 então o ciclo avançará imediatamente para a iteração seguinte sem que a saída digital 13 seja ativa.

- CONTINUE

A declaração da condição *continue*, ao contrário da condição de *break*, tem como função fazer avançar todo o ciclo para a próxima repetição do *loop*, ou seja, podemos fazer com que as funções no interior do ciclo não sejam executadas sempre que uma condição se verifique. No exemplo seguinte podemos ver uma de declaração da condição *continue* aplicada a um ciclo *for:*

```
for (int i=0; i<10 ; i++){
  if (analogRead(A0)<200) continue;
  else{
    digitalWrite(13,HIGH);
  }
}
```

O que acontece neste exemplo é que sempre que é verificada esta condição o ciclo é interrompido e avança para a iteração seguinte, não sendo totalmente executado o conjunto de instruções do ciclo.

Funções digitais

O código de Arduíno tem a particularidade de apresentar um conjunto de funções que interagem diretamente com as suas entradas/saídas digitais e analógicas. Vamos agora abordar o conjunto de funcionalidades que devemos ter presentes quando vamos utilizar as portas digitais da placa.

Vamos começar por introduzir a seguinte função:

```
pinMode()
```

Esta função é usada para definir o tipo da porta digital, isto é, se esta irá funcionar como entrada ou saída (INPUT/OUTPUT).

Quando um destes pins é estabelecido como saída podemos então usar a função seguinte:

```
digitalWrite(2,HIGH)
```

Nesta função são efetuadas duas declarações em que a primeira é o número do pin e a segunda é o estado do pin sendo que as portas digitais vão de 0 a 13 e os estados

apenas podem ser HIGH ou LOW. Quando estabelecido o estado HIGH a porta passa a 5 VDC e quando no estado LOW a conexão é estabelecida com a terra (ground).

Quando o pin é estabelecido como sendo de entrada então podemos usar a seguinte função:

```
digitalRead(2)
```

Esta função tem como capacidade a verificação do estado em que se encontra a porta digital, indicando-nos o estado de cada uma das portas digitais em questão.

Funções analógicas

O sinal analógico tem como principal característica a sua variação suave entre os dois limites de tensão podendo adquirir qualquer valor entre os dois extremos, ao contrário do sinal digital que apenas tem a capacidade para permutar entre dois estados. No caso do sinal analógico, no intervalo 0-5V estão presentes um conjunto de 1024 valores que podem ser lidos, em que 0 corresponde aos 0V e 1023 aos 5V, isto é, temos uma resolução de 10 bits na leitura das portas analógicas. É importante notar que na maioria dos Arduínos as portas analógicas funcionam apenas como entrada (INPUT), característica verificada pelo Arduíno usado no projeto. No total o Arduíno possui 6 entradas analógicas nomeadas de A0-A5.

Como o Arduíno não possui a capacidade de envio de verdadeiros sinais analógicos este tem a capacidade de envio, através das portas digitais, de um tipo de sinal chamado impulso com modulação (pulse with modulation – PWM) que se baseia no ligar e desligar da saída variando a sua frequência. Na Figura 13 podemos melhor observar o tipo de processo descrito no que diz respeito à aproximação de um sinal digital num sinal analógico.

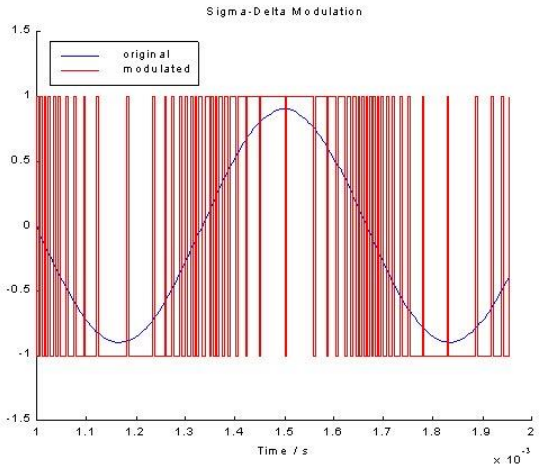

Figura 13: Esquema da modelação de um sinal por impulsos(Pohlmann, 1995)

No código de Arduíno a forma mais prática de interagir com as entradas analógicas é através da função:

```
analogRead(A0)
```

Usando a função anterior podemos efetuar uma leitura do valor de tensão registado na porta analógica com uma resolução de 10 bits (0-1023). O valor que é retornado pela função é portanto um valor no intervalo 0-1023 e uma vez que a entrada tem os limites de tensão de 0-5V então será necessária a simples conversão do valor em bits para o correspondente em tensão.

Funções do tempo

Uma vez que o Arduíno é um microcontrolador relativamente rápido são necessários por vezes alguns temporizadores para que se torne possível executar algumas das capacidades que pretendemos, tal como a execução das tarefas espaçadas ou os intervalos de tempo específicos.

Podemos começar por introduzir o *delay()* que tem como responsabilidade a paragem do programa sempre que exista a presença desta função. Um exemplo de um atraso de um segundo tem a seguinte representação:

```
delay(1000);
```

Esta função apresenta as suas interrupções em intervalos de milissegundos e os valores introduzidos no tempo terão obrigatoriamente de ser inteiros e positivos, daí que no exemplo anterior o valor de 1000 represente uma interrupção de um segundo.

Outro tipo de interrupção que podemos usar é:

```
delayMicroseconds(1000);
```

Esta última executa o mesmo tipo de funções que a função *delay(),* no entanto o que as diferencia é que nesta o tempo indicado é em microssegundos e não em milissegundos. Logo neste último caso com a indicação do tempo 1000 o que temos é uma interrupção de 0.001 segundos.

Continuando no que diz respeito a funções do tempo, temos agora uma função que apresenta características bastantes diferentes das duas anteriores, e que se trata da função:

```
millis()
```

Esta função vai ter a utilidade de funcionar como um cronómetro sempre que é chamada, correndo continuamente a partir do momento em que é executada no *loop*. No microcontrolador do Arduíno existem três timers diferentes pelo que podemos utilizar no máximo os três em simultâneo. Este tipo de função pode ser usado, por exemplo, como marco do tempo desde o início de uma determinada função. Esta função, ao invés do uso da função *delay(),* não provoca uma interrupção temporal em todo o código. A partir do momento que é iniciada a contagem, esta tem uma capacidade de armazenar até um número de 32 bits, o que equivale a cerca de um mês e meio até atingir o seu valor máximo e iniciar de novo a contagem a partir do zero.

Existe também a função:

```
micros()
```

Esta última executa o mesmo tipo de funções que a função *millis()* em que a única diferença é que esta efetua a contagem em intervalos de tempo de microssegundos.

Construção de funções

Todo o tipo de funções que vimos até agora são partes integrantes e já definidas pelo código de Arduíno, no entanto fora do *loop()* podemos também criar funções que irão posteriormente ser executadas ciclicamente no *loop()* ou mesmo no *Setup()* sempre que forem chamadas. O uso deste tipo de funções permite ao utilizador tornar mais organizado e simples o seu código e até por vezes uma redução do espaço ocupado em memória quando esta é executada mais que uma vez no mesmo ciclo passando a ser escrita apenas uma única vez e apenas chamada sempre que necessário. O uso de funções permite-nos assim uma leitura mais eficaz do código e a procura de possíveis erros mais simples.

Para a declaração de uma função é necessário a atribuição de um nome e estabelecer os parâmetros que irão ser passados para a mesma. Um exemplo simples de declaração de uma função é o seguinte:

```
void testFunction(){
}
```

Sempre que pretendermos usufruir desta função teremos de usar o seu nome *testFunction(),* para que esta seja executada. O uso desta função, por exemplo, no *loop()* terá o seguinte aspeto:

```
void loop(){
testFunction();
}
```

Outro tipo de funções consiste no tipo que retorna um valor após a sua execução e pode ser observado um exemplo nas linhas de código seguintes.

```
void testFunction(int num){
  int count=0;
  for (int i=0; i<num ; i++ ){
    if (analogRead(A0)>200) count++;
    delay(2000);
  }
  return count;
}
```

Esta última função contém o uso de um parâmetro e é do tipo que retorna um valor sempre que executada. Esta função efetua a leitura da entrada analógica zero com

intervalos de dois segundos um número de vezes que é definido através do parâmetro a introduzir na mesma.

Memórias do Arduíno

As capacidades de memória de um Arduíno podem ser vistas como uma grande limitação no uso do mesmo para o desenvolvimento de estruturas e projetos mais complexos. Durante todo o processo de programação de um Arduíno devemos sempre ter em conta a racionalização do espaço em memória para que este no final tenha a capacidade de albergar e desempenhar as funções do projeto que pretendemos desenvolver. Durante a construção de qualquer tipo de projeto se não existir esse cuidado facilmente nos deparamos com problemas associados às limitações de memória do Arduíno.

Basicamente no Arduíno existem três tipos de memória, *Flash*, *SRAM*, *EEPROM*. Na Tabela 6 podemos ver uma melhor e mais simples representação e explicação dos três diferentes tipos de memória. O tamanho reservado para o uso de cada uma das memórias está referenciada, a título exemplo, ao Arduíno UNO. As capacidades de armazenamento variam de acordo com o modelo de Arduíno que usarmos tal como pudemos ver na Tabela 1.

Tabela 6: Diferenciação geral dos tipos de memórias

Memória	Tamanho (bytes)	Tipo de armazenamento
Flash	32,768	Não volátil
SRAM	2048	Volátil
EEPROM	1024	Não volátil

Flash – Esta memória é responsável pelo armazenamento de todo o código, e bibliotecas usadas no mesmo, pelo que deve ser o mais compacto possível. Para o caso do microcontrolador ATmega328 (usado no Arduíno UNO) 500 bytes dos 32,728 bytes são ocupados pelo *bootloader (*é o programa que permite a comunicação do microcontrolador através da porta serial). Como se trata de tipo de memória não volátil esta mantém-se mesmo quando o Arduíno é desligado da corrente.

SRAM (*Static Random Acess Memory*) – É nesta memória que o microcontrolador acede, controla e manipula todas as variáveis e são executadas as instruções, enquanto

o microcontrolador está ligado à corrente. Contendo um número elevado de variáveis facilmente se atinge o limite de capacidade de utilização disponível, que consequentemente leva a problemas na execução das funções. Se a memória falhar o programa falha das mais variadas formas e simplesmente não funciona ou então pode funcionar mas não executando corretamente as suas funções.

As formas de verificar e prevenir alguns dos erros que possam ocorrer são as seguintes:

- Se o upload do programa for executado num computador podemos tentar transferir alguns dados e cálculos para o computador.

- No caso de existirem tabelas ou *arrays* muito grandes no código, sempre que possível, devemos tentar reduzir ao máximo o espaço ocupado pelas mesmas.

EEPROM (Electronically Erasable Read-Only memory) – A principal característica desta memória reside no fato de ser não volátil e com um tempo de vida bastante elevado. Esta memória permite assim guardar informação de variáveis e parâmetros que são medidos ao longo do tempo, de forma que não fiquem perdidos aquando de uma falha na alimentação do Arduíno. Esta memória interna do AVR tem um tempo de vida limitado a 1,000,000 de escritas e um número ilimitado de leituras. É um tipo de memória que funciona por endereços o que torna o seu uso um pouco mais elaborado que o normal no que diz respeito aos processos de escrita e leitura dos dados. (Camera, 2013)

Problemas com a RAM

Durante a programação de projetos mais complexos utilizando o microcontrolador ATmega328p facilmente nos deparamos com limitações da RAM. Uma das formas de reconhecer quando ocorrem erros na RAM é o aparecimento de mensagens estranhas na porta de série, ou o reiniciar automático do código comprometendo o funcionamento correto do equipamento. Quando acontecem erros deste tipo estamos com um problema de RAM e partes da RAM serão reescritas sobrepondo outras ou escritas em sítios incorretos. (Romeral, 2014)

Existe um conjunto de técnicas que podemos utilizar para conseguir libertar espaço da RAM, e as principais são as seguintes:

- Todas as *strings* a serem escritas na porta de série serão guardadas na RAM, ocupando espaço desnecessariamente, podemos então usar a função F() ou o atributo PROGMEM para que sejam armazenadas na memória Flash libertando assim espaço na RAM.

- A construção dinâmica de *strings* ocupa muito espaço na memória do microcontrolador, pelo que devemos recorrer ao uso da função reserve() para conseguir reduzir o uso de espaço. Para um melhor entendimento desta função vamos descrever a construção de uma *string* e o respetivo espaço ocupado pela operação.

Consideremos a seguinte construção:

```
String myString;
myString = "i=";
myString += someInteger;      // value 1234
myString += ", is that ok?";
```

A construção de uma simples *string* vai ocupar 3 bytes da memória temporária. Com a escrita de "i=" a *string* vai reservar mais 3 bytes para essa mesma escrita e serão destruídos os bytes temporários libertando os primeiros 3 bytes usados, e ficamos assim com uma memória de 3 bytes livres e 3 bytes usados.

Na segunda linha o valor inteiro será convertido num objeto temporário com o tamanho de 5 bytes. Para a operação de acréscimo serão necessários 7 bytes que irão ser posicionados a seguir aos 5 bytes da memória temporária. Temos agora que a soma total de bytes é: 3+3+5+7.

Na terceira linha é repetido o processo, em que serão colocados na memória temporária 14 bytes e a *myString* será alargada para 27 bytes do final, ficamos assim com um número total de bytes: 3+3+5+7+14+27. Temos assim que os 27 bytes finais vão estar colocados a seguir a um conjunto de 32 bytes de espaço vazio necessário na construção da *string*.

```
String myString;
myString.reserve(26);
myString = "i=";
myString += someInteger;      // value 1234
myString += ", is that ok?";
```

Usando esta função, no início, vai ser criado um conjunto de 27 bytes de memória que pode ser usada na construção das *strings*, limitando-a antes da próxima linha.

Micro cartão SD

Uma outra capacidade embutida em alguns dos equipamentos disponíveis é o recurso a um cartão micro SD para armazenamento de grandes quantidades de dados que de outra forma ocupariam demasiado espaço na memória RAM do Arduíno. A designação de SD significa *secure digital flash memory*.

Existem várias opções no mercado disponíveis para aplicação do micro SD a um equipamento Arduíno, no entanto a que é de mais fácil uso e acesso é a que está incluída no *Arduino Ethernet shield* que já traz todo o *hardware* necessário para explorar todas as capacidades desta memória externa. O monitor série pode ser usado como interface de verificação de estado dos processos a ocorrerem no SD, podemos assim ter a informação se um ficheiro foi acedido ou não ou então se um registo foi ou não guardado com sucesso. O dispositivo de Ethernet está ligado ao pin 10, por definição, e o SD está ligado ao pin 4. Desta forma estes pins não poderão ser usados para outras funções.

RTC – "*Real Time Clock*"

O RTC é mais um acessório disponibilizado pelos fabricantes do Arduíno que tem como finalidade fornecer a informação da hora ao equipamento continuando a sua contagem mesmo quando este se encontra desligado da corrente. Torna-se assim uma ferramenta muito eficaz para a criação de projetos que necessitem de acesso a informação temporal após a ocorrência de falhas de energia. Este tipo de aplicação permite ao Arduíno ter um sistema preciso de relógio libertando assim a memória do CPU dos cálculos dos dados da hora permitindo uma menor ocupação de espaço na memória do programa.

O RTC utilizado é do tipo DS1307 que contém um contador integrado BCD ("*Binary Coded Decimal*") de baixa energia com capacidade para contar segundos, minutos, horas, dias, meses e anos equipado com 56 *bytes* de memória RAM estática não volátil. A informação sobre o tempo e data é colocada num registo especial e transferida para o microcontrolador do Arduíno via I2C (também designado de *two wire interface*). A

transferência de dados com o Arduíno utilizado (*Duemilanove*) é efetuada através dos sinais SCL e DAS que estão disponíveis nos pins A4 e A5.

Para o seu funcionamento de operação normal necessita de uma fonte de energia de 5 Vdc e apresenta um consumo operacional de 1.5 mA e em modo de bateria um consumo de 500 nA.

Comunicação TCP vs UDP

Existem dois tipos de protocolos de internet para o trafego de dados, que são o protocolo TCP (*transmission control protocol*) e o UDP (*user datagram protocol*). É de salientar diferenças significativas entre estes dois protocolos sendo que comparativamente apresentam vantagens e desvantagens de acordo com as características que pretendemos ver cumpridas na comunicação que vamos efetuar. De seguida são apresentados um conjunto de características representativas de cada uma das comunicações. (wikibooks, 2014)

Comunicação TCP

- É um protocolo de comunicação que requer a existência de uma conexão estabelecida para que seja permitida a troca de dados.
- Tem como função o envio de uma mensagem através da internet de um computador para outro, tal como uma chamada telefónica exige que seja estabelecida para que se troquem dados.
- É direcionada para aplicações que exijam uma grande fiabilidade na transmissão dos dados sem a preocupação no fator velocidade e transmissão.
- É usada por outros protocolos como HTTP, HTTPs, FTP, SMTP, Telnet.
- Possibilita a garantia absoluta na transferência de dados e que os dados chegam ao destino seguindo a ordem pela qual foram enviados.
- Os dados são lidos como um fluxo de bytes e não são transmitidas indicações sobre os limites da mensagem.
- Faz o controlo de fluxo e requer a existência de três pacotes para que a conexão seja estabelecida antes dos dados serem enviados.

- Faz uma verificação de erros no envio dos dados sendo fornecida essa informação sempre que ocorre algum problema na transmissão dos dados ou na conexão.
- A entrega de todos os dados é gerenciada e sempre que ocorra perda de dados estes são retransmitidos.

Comunicação UDP

- Este tipo de comunicação de dados não exige a existência de uma conexão estabelecida entre duas pontas.
- É um protocolo usado no transporte e transferência de mensagens, uma vez que não é baseado na necessidade de conexão constante e sendo assim a partir do momento que o emissor envia todas as mensagens a conexão termina, não confirmando a entrega final dos dados.
- Revela eficiência sobretudo para aplicações que necessitem de velocidade e eficiência na transmissão dos dados, como no caso dos jogos *online* e ainda por servidores que respondem pequenas *queries* a um grande número de clientes.
- É usada por protocolos como DNS, DHCP, TFTP, SNMP, RIP, VOIP.
- Os pacotes enviados são independentes entre si, pelo que não é necessariamente respeitada uma ordem na transmissão e entrega dos dados. Isto é, dados enviados depois podem alcançar um mesmo destino primeiro no caso de tomarem um caminho mais curto.
- Esta comunicação é mais rápida que a TCP porque não existe verificação de erros.
- Não garante que as mensagens cheguem ao seu destino.
- Os pacotes são enviados individualmente e a sua integridade é apenas verificada se estes chegarem ao destino, que terá de enviar uma resposta de volta a comunicar que ocorreu algum erro na transmissão.
- Não tem opção de controlo de fluxo dos dados e faz uma análise de erros na mensagem mas no entanto não tem opções de resposta no caso de algum erro ser detetado.

Se tivermos em conta o conjunto de características evidenciadas e que identificam cada tipo de comunicação podemos fazer a pergunta do porquê de se usar UDP se a comunicação TCP apresenta inúmeras vantagens em comparação. Mas no entanto se tivermos em conta, por exemplo, o caso de jogos online a quantidade de dados trocada é muito grande e o facto da comunicação TCP exigir a entrega e ordem na troca de dados facilmente pode provocar um congestionamento da banda o que permite concluir que nem sempre o TCP é o mais adequado. A comunicação UDP é assim mais direcionada para casos que são sensíveis à velocidade na troca de dados, enquanto o TCP para casos em que seja essencial a garantia na transferência e entrega de dados entre os diferentes pontos.

A escolha do protocolo de comunicação irá ser sempre dependente das situações e limitações que pretendemos para cada caso, visto que para diferentes situações podemos ter um ou outro protocolo como mais vantajoso de usar. (Diffen, 2014)

Escolha de sensores

A variedade de sensores atualmente disponibilizados no mercado é suficientemente alargada ao ponto de já existir uma grande variedade que pode interagir diretamente com os microcontroladores embutidos nos equipamentos Arduíno. O que procuramos é um tipo de sensor capaz de efetuar leituras de parâmetros físicos e traduzir essas mesmas leituras para um sinal de corrente ou tensão que pode ser recolhido e analisado utilizando equipamento Arduíno.

Um tipo de sensores que podemos utilizar são sensores de infravermelhos que têm como funcionalidades a deteção das variações de distanciamento entre o sensor e objetos que sejam colocados até ao seu limite de alcance. Podem ser também usados sensores de áudio que têm como funcionalidades, por exemplo, a deteção da presença de algum movimento humano ou de sons em geral para a aplicação de uma posterior ação.

No entanto os tipos de sensores que a que vamos dar particular interesse são sensores de temperatura, pressão, tensão, fluxo de corrente e outros tipos que sejam frequentemente utilizados em medidas de parâmetros físicos numa indústria.

Um aspeto importante, comum a todos os sensores que vamos utilizar, será a transmissão de uma resposta em corrente às alterações das condições medidas pelos respetivos sensores. É neste seguimento de ideias que surgem os sensores de corrente

em circuito que podem ser adaptados para tensão com o recurso a uma montagem com as resistências adequadas. (bapihvac, 2014)

Os sensores utilizados para o desenvolvimento do projeto foram do tipo de corrente de 4-20mA, que serão mais à frente descritos.

Introdução histórica

Este tipo de sensores tem a sua origem datada por volta da década de 50 com a chegada dos circuitos elétricos e eletrónicos, onde os sinais padrão de 4-20mA se afirmam como um dos meios mais populares para transmissão e controlo eletrónico em ambientes industriais.

Antes da adoção dos sistemas de controlo eletrónico os edifícios recorriam a sistemas de controlo pneumáticos onde grandes compressores enviavam sinais pneumáticos de 3psi a 15psi que seriam controlados por válvulas pneumáticas controladoras com o objetivo de enviarem um sinal proporcional aos atuadores e controladores através do edifício. A pressão de 3psi representaria o valor de zero e a de 15psi o valor máximo de 100%. Qualquer valor de pressão inferior a 3psi representaria uma condição de alarme.

É neste seguimento de ideias que surgem os sensores do tipo 4-20mA baseados em circuitos controlados de corrente que usam 2 fios transmissores para converter os diferentes sinas processados à entrada pelo sensor em corrente continua com o objetivo de ser capaz de transmitir o sinal ao longo de alguma distância com perdas muito reduzidas.

Sensores baseados em circuitos de corrente

Para melhor compreendermos este novo conceito vamos recorrer ao simples triângulo que está apresentado na Figura 14.

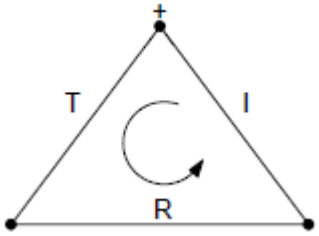

Figura 14:Circuito simbólico de sensor de corrente (Acromag)

Através do triângulo podemos considerar os três componentes comuns de um circuito de corrente, a forma como estes estão relacionados entre si e até a direção do fluxo de corrente, onde cada lado do triângulo representa um componente e os vértices à ligação entre eles.

Aceitando a convenção de que o fluxo de corrente se desloca no sentido da fonte positiva para a negativa assume-se, tal como representado na figura, que a corrente se irá deslocar no sentido contrário aos ponteiros do relógio.

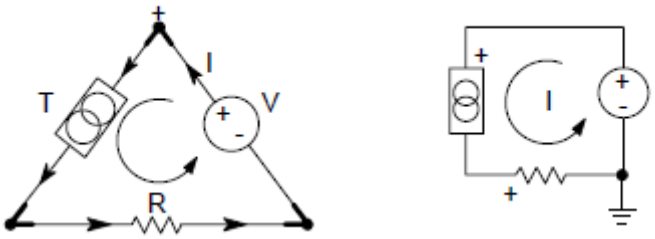

Figura 15:A figura da esquerda mostra o circuito com a introdução dos componentes, e a figura da direita um circuito de corrente tradicional

Na Figura 15 do lado direito podemos ver o exemplo tradicional de um circuito de corrente tipicamente de 4-20mA. No nosso caso em particular o controlo é efetuado a partir das entradas analógicas da placa de Arduíno.

A letra “T” corresponde ao transmissor, a letra “R” ao recetor e a letra “I” corresponde à fonte de corrente ou outra fonte de energia. A corrente tem origem na fonte de energia e flui de forma controlada através do elemento transmissor passando pelo elemento recetor onde vai ser gerada uma tensão que é facilmente medida por uma entrada analógica de um controlador ou por um dispositivo de monitorização.

O transmissor tem a capacidade de transmitir dados provenientes de um sensor através de um circuito de corrente de dois fios, sendo que num circuito deste tipo apenas pode haver uma saída transmissora. A real função de um transmissor consiste na transformação de sinais como fluxo, pressão, temperatura, humidade, entre outros, provenientes do mundo real no sinal de controlo necessário para regular o fluxo de corrente no circuito, onde o nível do circuito será ajustado pelo transmissor de forma a ser proporcional ao sinal de entrada do sensor real. É importante referir que a saída do sinal transmissor tipicamente de 4mA para representar o valor zero de entrada correspondente a 0% e a saída de 20mA representa a escala máxima de calibração do

sinal de entrada, correspondente ao valor máximo ou 100%, tal como está representado na Figura 16.

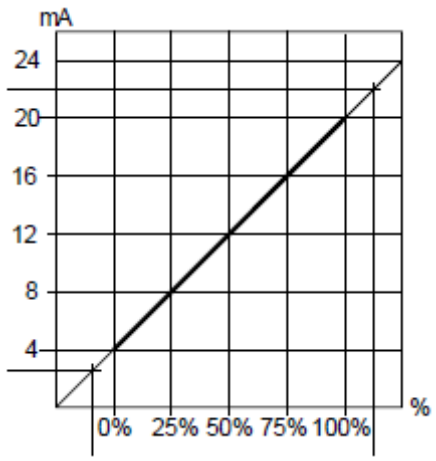

Figura 16: Representação gráfica da linearidade de um sensor de 4-20mA

Baseado em toda a descrição anterior podemos facilmente perceber que para a definição de um sensor de 4-20mA são necessários atribuir dois parâmetros: um valor mínimo que corresponderá ao caso em que o valor da saída do transmissor é de 4mA e um valor máximo que irá corresponder ao valor a atribuir à leitura do sensor quando o transmissor regista uma saída de 20mA. Este modo de funcionamento tem como base a hipótese de que a resposta em corrente do sensor varia sempre linearmente com a variação da grandeza física em medição.

O valor mínimo do transmissor 4mA permite distinguir os casos em que um sensor está a medir o valor zero dos casos em que o valor zero do sensor indiciam uma avaria ou uma falha de energia, permite assim uma maior robustez e fiabilidade no registo das leituras efetuadas por este tipo de sensor.

Desenvolvimento do projeto de deslastre

O conceito de deslastre que se pretende desenvolver com o equipamento Arduíno trata-se de um tipo de sistema capaz de monitorizar uma série de parâmetros físicos e efetuar

uma ação de resposta de acordo com um conjunto de condições impostas, que podem ser definidas pelos utilizadores. O desenvolvimento do mesmo terá de ser efetuado por etapas em que são atingidos objetivos que irão estabelecer aos poucos bases sólidas para que no final exista a possibilidade de obtenção de um sistema robusto passível de ser instalado com confiança numa instalação industrial.

Um dos ideais no desenvolvimento do projeto será o aproveitamento máximo das capacidades disponibilizadas pelo equipamento Arduíno para que se consiga a obtenção de bons resultados usando um equipamento económico e com características de memória limitadas. Assim sendo iremos recorrer ao uso de todas as entradas analógicas presentes no modelo utilizado (*Arduíno Duemilanove*) para a ligação dos sensores que se pretende controlar e recorrer às portas digitais para o envio de ações aos equipamentos a controlar. Irá também ser usada uma placa de Ethernet que funcionará como meio para a construção de um servidor web onde será desenvolvida uma página totalmente destinada ao controlo e visualização de todos os processos e parâmetros disponíveis no projeto final.

De uma forma simplista o produto de deslastre pode ser descrito como um equipamento ao qual será enviada uma ordem, através de uma equação, em que o sistema será capaz de interpretar de forma inteligente essa informação e consequentemente executar ações.

Estrutura do projeto

No início foi efetuado um levantamento prático e objetivo das necessidades básicas de um sistema típico de deslastre de cargas em ambiente industrial, isto é, que tipo de funcionalidades o Arduíno terá que resolver para cumprir com o esperado. Pretende-se que o equipamento proporcione ao utilizador flexibilidade na criação de uma variedade de comandos de controlo para que a sua adaptação à realidade se efetue de forma a cumprir o maior número de necessidades possíveis.

Numa fase inicial pretende-se a estruturação de um código com a capacidade para a leitura e interpretação de parâmetros físicos recolhidos por sensores que serão conectados ao Arduíno recorrendo às capacidades das suas portas analógicas.

Um passo importante no desenvolvimento do código será a criação de uma plataforma que permitirá a interação do utilizador com o equipamento tornando-o numa plataforma programável e adaptável às necessidades do utilizador. A forma mais eficaz encontrada para solucionar o problema foi a criação de um servidor web que servirá como painel de interação com o utilizador.

Leitura de sinais analógicos

O início da construção do equipamento terá que se basear numa rotina que permita a leitura de parâmetros físicos enviados por sensores. Tal como foi discutido anteriormente serão utilizados sensores de corrente pelo que na construção final do equipamento será necessário o recurso a resistências para a conversão do sinal para tensão. Para que seja possível uma leitura em tensão do sinal de corrente terá de ser montado um circuito que recorrerá ao uso de uma resistência, de 250Ω, em série com o sinal de corrente para transformar o sinal em tensão de valor máximo limitado a 5V. A conversão é feita pela seguinte equação:

Equação 1

$$V_{max} = 5V$$

Equação 2

$$I_{max} = 20\ mA$$

Equação 3: Lei de Ohm

$$V = RI$$

Facilmente concluímos que o valor da resistência a usar será de:

Equação 4

$$R = \frac{V_{max}}{I_{max}} = 250\ \Omega$$

Depois de calculado o valor da resistência será necessário recorrer a uma montagem de um circuito elétrico simples, tal como o da Figura 17, para dar início aos primeiros testes de leituras provenientes de um sensor real. (Industrologic, Inc., 2014)

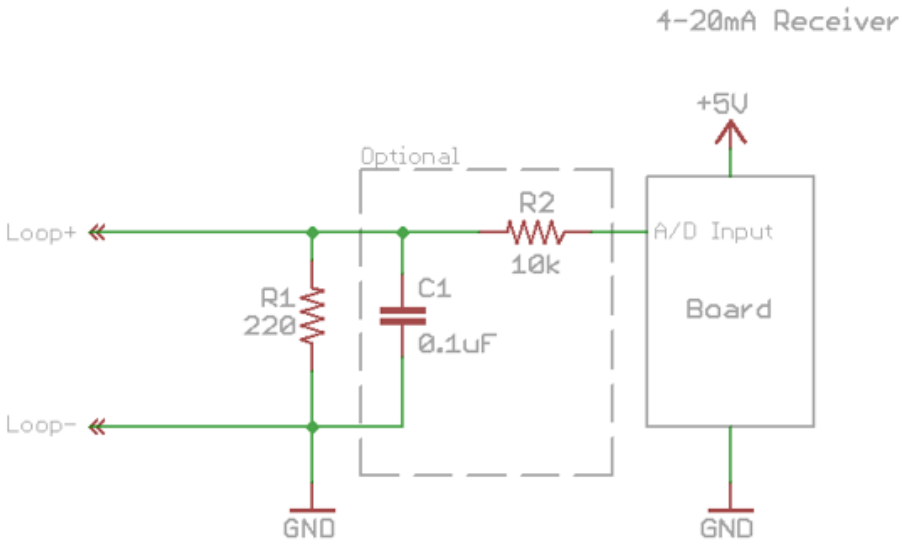

Figura 17: Montagem do circuito elétrico para conversão de 4-20mA para 0-5V

Uma das caraterísticas importantes no código desenvolvido está na capacidade de armazenamento constante do valor recolhido pelas entradas analógicas, sendo este valor armazenado num *array* associado a uma variável global que irá ser responsável pelo transporte da informação para as diferentes funções do código.

Cada entrada analógica efetua uma leitura enviada pelo sensor para a porta mas este é reproduzido na forma de um valor de 10 bits, ou seja, sempre que é efetuada por exemplo a leitura de um sinal de 5V o valor correspondente reproduzido pelo Arduíno é 1023. Desta forma as leituras das entradas analógicas vão encontrar-se dentro de um alcance de 205-1023, significando que este alcance terá de ser trabalhado para que apresente ao utilizador dados num formato enquadrado com a realidade do sensor analisado.

Para todos os sensores de 4-20mA usados no projeto, idealmente, considera-se que respondem linearmente às variações registadas nas suas leituras em que uma resposta de 4mA corresponderá ao valor mínimo que o sensor consegue alcançar e o valor de 20mA corresponde ao valor máximo de alcance do sensor. Desta forma será essencial a caraterização individual dos sensores atribuindo os seus valores característicos mínimo e máximo que em geral são apresentados na documentação técnica de cada um deles.

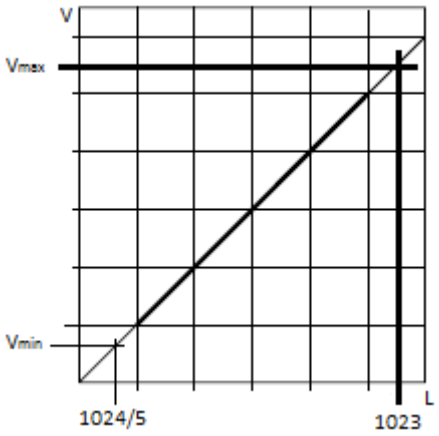

Figura 18: Linearização da leitura proveniente de um sensor de 4-20mA no Arduino

Recorrendo à representação gráfica da linearização de uma leitura analógica, por exemplo Figura 18, proveniente de um sensor responsável pela recolha de um parâmetro físico podemos chegar à equação que descreve a tradução do valor analógico para um valor adaptado a cada sensor:

$$V = \frac{(V_{max} - V_{min})}{\frac{4}{5} \times 1023} \times L + V_{max} - \frac{5}{4} \times (V_{max} - V_{min})$$

V_{max} – Representa o valor característico do sensor correspondente ao máximo de corrente de 20mA

V_{min} - Representa o valor característico do sensor correspondente ao mínimo de corrente de 4mA

Realizada esta análise será assim muito importante a criação de uma interface que permita aos utilizadores inserirem os parâmetros correspondentes a cada sensor. E uma vez que o Arduino possui um total de 6 entradas analógicas serão desenvolvidos processos que permitem a configuração individual dos parâmetros máximo e mínimo para cada um dos 6 sensores que podem ser conectados ao equipamento.

Características do servidor web

A criação de um servidor web foi a solução encontrada para permitir a possibilidade de interação entre o utilizador e o equipamento. Recorrendo às capacidades apresentadas pela aplicação da placa de Ethernet no equipamento foi possível a criação de um

servidor que servirá como painel para a visualização e troca de dados entre o utilizador e o equipamento. Esta necessidade de interação torna-se evidente quando começamos a pensar na criação de um equipamento compatível para todos os diferentes tipos de sensores, desde que estes sejam do tipo de corrente de 4-20mA.

Durante o desenvolvimento do servidor estiveram sempre presentes como guias as diretivas de criação de uma plataforma simples e eficiente na ótica do utilizador, permitindo uma programação e observação dos estados dos sensores sem grande esforço de compreensão. Assim sendo uma plataforma deste tipo deverá possuir as seguintes capacidades:

- Possibilidade de configuração dos parâmetros máximos e mínimos característicos de cada sensor.
- A criação/alteração/remoção deverá ser sempre possível em qualquer momento pretendido pelo utilizador.
- Os sensores correspondentes a cada uma das portas analógicas deverão ser passiveis de atribuir à entrada analógica pretendida sem necessidade de qualquer ordem.
- Deverá estar sempre presente um painel com a informação das entradas que estão ativas, dos valores registados pelas mesmas assim como os correspondentes parâmetros atribuídos.

Depois de desenvolvida e otimizada a componente de leitura e monitorização de todos os sensores deve-se avançar para a fase seguinte que consiste na criação de processos que permitam ao utilizador o controlo sobre os sensores conectados.

Esta é a fase em que será desenvolvida toda a componente capaz de interpretar os valores que estão constantemente a ser recolhidos pelos sensores e interpreta-la de uma forma que permita ao utilizador programar o ativar/desativar de saídas digitais que efetuarão uma ação como resposta a uma ou mais condições. O desenvolvimento desta componente foi baseado na resolução.

- O equipamento terá de ter a possibilidade de envio de sinais de saída digitais
- O utilizador deve ter a possibilidade de enviar um conjunto de comandos de deslastre que irão interagir apenas com sensores que se encontrem ligados
- O comandos devem ser capazes de autonomamente ativar ou desativar uma saída digital sempre que verificada ou não uma condição

- Depois de programado o equipamento deverá funcionar como um painel, apresentando constantemente informação atualizada do estado das saídas digitais
- Deve existir a possibilidade de ativar ou cancelar cada um dos comandos de deslastre programados para qualquer das portas
- O utilizador terá o controlo sobre a apresentação das portas digitais e deverá ter a possibilidade de reprogramar cada uma delas sem limitações

Estrutura do servidor

Estabelecidas as diretivas gerais que se pretendem alcançar ao longo do desenvolvimento do projeto será importante pensar numa forma intuitiva de apresentação da informação básica necessária assim como os comandos de controlo.

Para transformar a plataforma acessível do ponto de vista do utilizador mais inexperiente apenas será apresentada a informação essencial para a programação dos sensores na perspetiva de simplificação na compreensão do sistema.

Seguindo as diretivas anteriores inicialmente a ideia baseava-se em criar uma plataforma constituída por um conjunto de botões que quando selecionados proporcionam ao utilizador a possibilidade de atribuição dos parâmetros. Neste formato todos os botões seriam listados e seria apresentada toda a informação dos sensores e portas digitais ativas. Um exemplo deste formato está apresentado na Figura 19. É importante todo este trabalho de configuração estética com programação HTML para que seja possível otimizar o funcionamento de transporte de informação pelo código.

Usando o formato apresentado na Figura 19 a visualização de informação por parte do utilizador fica bastante complicada e muito pouco intuitiva. Note-se que nesta visualização existe a possibilidade de criação que fica omitida sempre que configurado um sensor, e neste formato apenas é possível a configuração no máximo de três sensores que podem sempre ser reconfigurados ou cancelados como podemos observar pelos botões na imagem. Sendo que a configuração do sensor é feita definindo o tipo de comparação que se pretende efetuar e o valor com o qual se pretende efetuar a comparação.

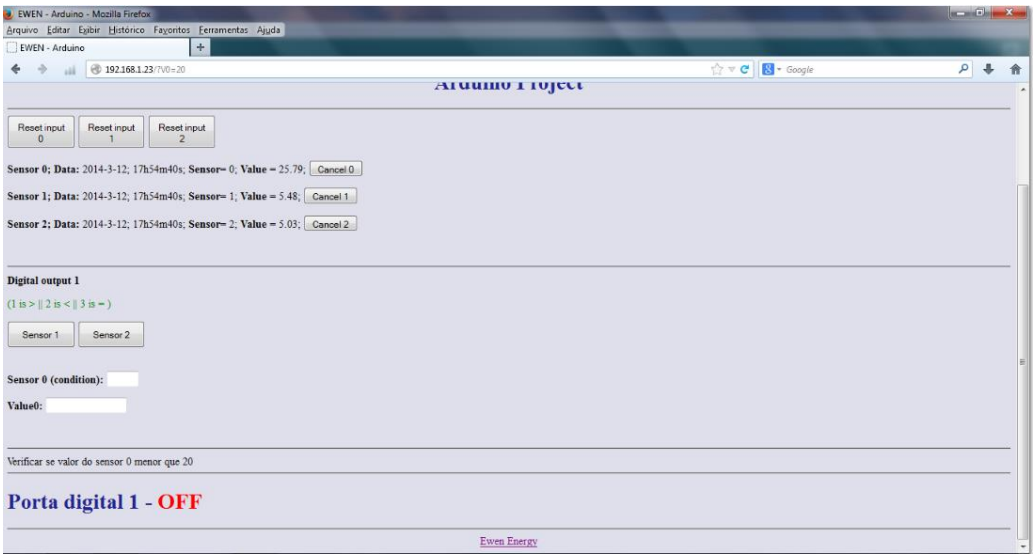

Figura 19: Formato de teste da apresentação do servidor Web

Este tipo de abordagem apresenta um grande conjunto de limitações que implicam uma evolução obrigatória da apresentação da informação assim como um aumento do número de entradas analógicas que podem ser configuradas pelo utilizador. O sistema deve também permitir um maior número de condições numa tentativa de tornar mais versátil o equipamento para adaptação aos mais variados casos que possam ocorrer nas instalações.

A tentativa que encontrei para tornar a plataforma eficiente e prática para o utilizador foi a criação de uma divisão por menus que permitirão a organização da informação. De um dos lados irão ficar todas as opções de configuração do sistema e do outro a informação sobre o estado dos sensores e das portas digitais ativadas ou desativadas. Um exemplo desta apresentação está ilustrada na Figura 20.

É importante de referir que o sistema poderá fornecer informação atual das leituras que estão a ser registadas pelo equipamento, possibilitando desta forma o funcionamento com ecrã de monitorização.

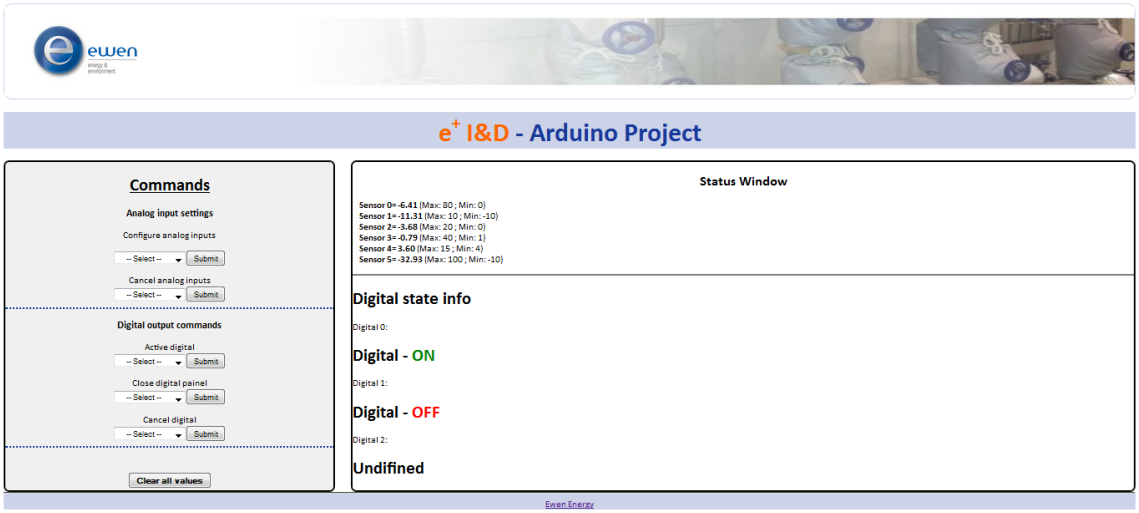

Figura 20: Representação do esquema final do sistema de deslastre

Descrição das funcionalidades do sistema de deslastre

Depois de apresentado o esquema geral do sistema de deslastre vamos começar agora por descrever todo o tipo de configurações que podem ser efetuadas no sistema e apresentar a divisão onde estão inseridas todas as possibilidade de configuração, recorrendo à Figura 21.

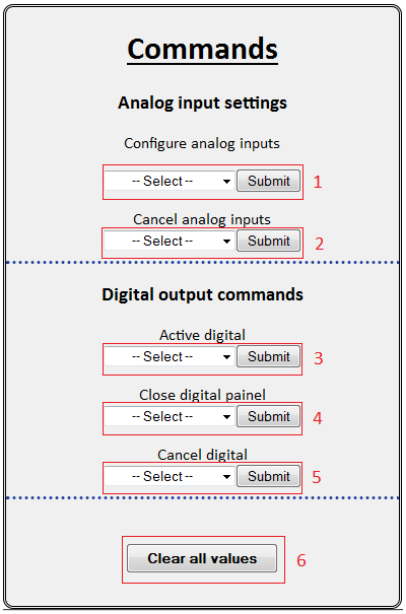

Figura 21: Apresentação legendada da divisão de menus

Recorrendo à numeração apresentada na Figura 21 temos:

1. Configuração das entradas analógicas

   É neste elemento que podemos efetuar a seleção da entrada analógica do equipamento ao qual foi conectado o sensor. Depois de efetuada a escolha da entrada são-nos apresentadas duas caixas de texto, como mostra a Figura 22, onde podemos definir os parâmetros característicos do sensor.

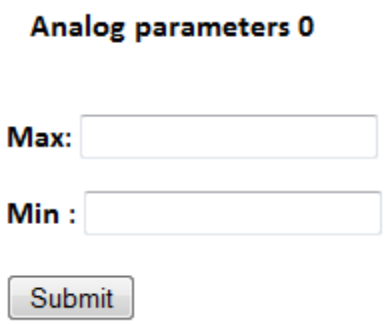

Figura 22: Definição dos parâmetros do sensor

   A seleção de uma entrada já em uso irá permitir ao utilizador a reconfiguração dos parâmetros do sensor, sendo que após este passo será sempre necessário o preenchimento dos dois parâmetros para que a atribuição se perpetue de forma correta. O equipamento não permite que o utilizador defina o parâmetro mínimo maior que o parâmetro máximo. Existe também um processo que não permite a introdução de parâmetros máximos e mínimos iguais.

2. Cancelar entradas analógicas

   Esta secção tem como principal funcionalidade o desativar de qualquer das entradas analógicas definidas. Assim usando esta função podemos em qualquer momento anular a ligação do sensor ao equipamento proporcionando ao utilizador grande facilidade na programação do equipamento.

3. Ativar as saídas digitais

   Depois de configurado pelo menos um sensor é já possível atribuir uma condição que quando verificada faz com que a porta digital seja ativada, e uma vez selecionado fica visível uma caixa onde é possível introduzir a equação, como podemos ver na Figura 23.

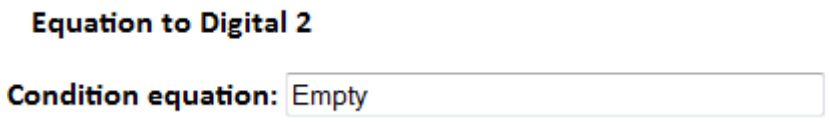

Figura 23: Espaço para introdução da equação de comando da porta digital

Para permitir o envio de uma ordem para o equipamento foi necessária a definição de um conjunto de termos padrão para que a equação escrita seja reconhecida e interpretada de forma a enviar os comandos pelas saídas digitais. Assim os sensores podem ser definidos na equação com os seguintes nomes:
Entrada analógica 0: S0, s0, Sensor0, sensor0;
Entrada analógica 1: S1, s1, Sensor1, sensor1;
Entrada analógica 2: S2, s2, Sensor2, sensor2;
Entrada analógica 3: S3, s3, Sensor3, sensor3;
Entrada analógica 4: S4, s4, Sensor4, sensor4;
Entrada analógica 5: S5, s5, Sensor5, sensor5;

O equipamento permite ao utilizador a comparação de qualquer uma das entradas com um valor real (ex: sensor3 <10), assim como a comparação entre diferentes sensores (ex: S2> Sensor3).
O conjunto de operadores que podem ser utilizados no equipamento é:
“>; <; minus; or; and”.

4. Fechar a porta digital

Depois de ativada e configurada uma porta digital o utilizador tem a opção de sair da vista de configuração para o painel principal permitindo-lhe ter uma perceção do estado geral de todas as portas.

5. Cancelar a porta digital

Esta funcionalidade permite ao utilizador o apagar a informação contida como condição da saída digital. Assim sempre que selecionado o cancelamento de uma porta digital esta perde a informação que tinha guardada e passa ao estado de desativada não efetuando qualquer tipo de função.

6. Limpar todos os valores

   Este botão é apenas acessível ao administrador (acesso garantido através de palavra passe) e tem como funcionalidade limpar todos os dados que o equipamento mantém guardados em memória para que em caso de falha de energia este tenha a capacidade de reestabelecer as suas funções normalmente sem exigir nenhuma ação por parte do utilizador.

Armazenamento de dados na EEPROM

Como já foi discutido anteriormente o único tipo de memória não volátil que o Arduíno possui é a memória EEPROM, e daí ser necessário retirar o máximo partido dela no desenvolvimento do projeto. A grande necessidade que se impõe na construção de um sistema autónomo baseia-se na capacidade que este tem de, uma vez configurado, operar normalmente mesmo quando ocorre alguma falha na alimentação do equipamento e daí ser necessária a criação de uma função que se iniciará sempre que o instrumento é reiniciado. As capacidades desta função são a atribuição dos antigos valores das variáveis globais armazenadas e o restabelecer das condições que estavam a operar no momento da falha de energia através da leitura dos dados guardados em posições específicas da EEPROM.

Conjunto de equações para programação do equipamento

A definição de uma plataforma com programação por equação surge com a necessidade de cruzamento da informação recebida pelo equipamento e posterior uso para efeitos de comparação com valores introduzidos pelo utilizador ou então entre os elementos sensitivos em medição. Assim basicamente o utilizador será capaz de introduzir um valor de acordo com a informação que é enviada por um sensor e registada pelo Arduíno.

Como foi falado anteriormente o equipamento tem capacidade para reconhecer diferentes tipos de comandos que quando colocados em conjunto vão construir uma condição lógica que irá despoletar uma ação por parte do equipamento.

Apresentados os diferentes comandos que temos disponíveis para comunicar ordens ao equipamento o conjunto de condições que se podem enviar ao mesmo são as seguintes:

- S1<10 / S1>10
- S1>S2 / S1<S2
- S1andS2>s0 / S1andS2<s0
- S1andS2>10 / S1andS2<10
- S1orS2>s0 / S1orS2<s0
- S1orS2>10 / S1orS2<10
- S1minusS2<S0 / S1minusS2>S0
- S1minusS2<10 / S1minusS2>10

As equações apresentadas nos pontos anteriores representam um conjunto exemplificativo dos diferentes comandos que podem ser enviados para o equipamento. É de notar que a conjugação e permutação de sensores e de valores é completamente livre onde qualquer valor inteiro pode ser introduzido para efeitos de comparação.

Se forem introduzidos dados no equipamento que não sejam compatíveis com estes formatos de equações ou então se houver um erro na escrita da equação o instrumento vai enviar um alerta ao utilizador com a seguinte mensagem de erro: “EQUATION ERROR”. E neste caso não será efetuada qualquer ação sob a porta digital em questão, mantendo-se a saída digital isenta de responsabilidades no controlo de qualquer porta.

Descrição da informação na janela de estados

Feita uma apresentação de todas as funcionalidades que podemos ver resolvidas usando este tipo de sistema pode-se fazer uma descrição do tipo de informação apresentada na janela de estados aos utilizadores sobre as configurações que foram efetuadas.

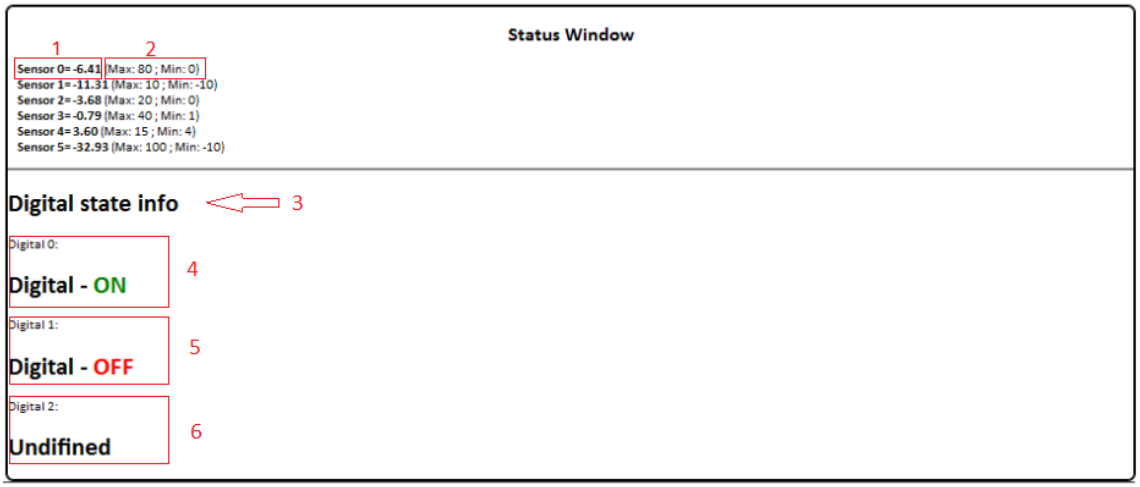

Figura 24: Representação da janela de estados do sensor

Para uma análise mais facilitada da janela de estados vamos recorrer à numeração descrita pela Figura 24, e assim sendo temos:

1. Esta primeira seleção apresenta informação do valor que está a ser lido pelo sensor em tempo real. A informação deste valor é importante para facilitar e tornar mais percetível a definição das condições pelos utilizadores
2. Neste ponto são apresentados os respetivos parâmetros máximo e mínimo atribuídos ao sensor em questão para permitir ao utilizador detetar a existência de erros na configuração
3. Nesta segunda divisão são apresentadas todas as informações correspondentes às configurações das saídas digitais
4. Corresponde à descrição do estado da saída digital 0
5. Corresponde à descrição do estado da saída digital 1
6. Corresponde à descrição do estado da saída digital 2

É importante referir que caso o utilizador necessite saber qual a condição imposta a uma das saídas digitais apenas terá de ativar a visualização da mesma sendo visível o comando definido para a saída em questão. Esta informação apenas não está visível no painel principal por uma questão de organização para que a informação visível não se torne excessiva e dificulte o acesso a informação realmente útil.

Proteção com palavra passe

Sendo este um equipamento que pode ser ligado a uma rede local e acedido por diferentes utilizadores é importante a criação de uma proteção com vista a restringir o acesso apenas a um número reduzido de passos. Assim desta forma foi criada uma palavra passe obrigatória que será necessário inserir na plataforma para que seja possível o acesso à página de configuração do sistema. Pode ser visualizado na Figura 25 a imagem que é representada sempre que um utilizador acede pela primeira vez à pagina do equipamento.

Figura 25: Esquema de introdução da palavra passe

Uma vez que existe no sistema um botão que elimina todos os valores armazenados na memória foram criadas duas palavras passe com permissões diferentes que são disponibilizadas ao utilizador permitindo-lhe liberdade na atribuição de acessos com permissões diferentes.

“admin” – permite o acesso a todas as configurações do sistema.

“user” – este utilizador não tem permissão para eliminar todos os dados da memória, mas tem a possibilidade de configurar todo o restante sistema.

Este tipo de mecanismo de segurança permite aumentar a robustez do equipamento adicionando um argumento forte para o seu lançamento e para a sua preferência no ato de utilização por parte do utilizador.

Construção de caixa para o equipamento

Depois de desenvolvida toda a componente de programação o próximo passo consistiu na construção de uma envolvente física que permita a passagem de um equipamento de teste para um passível de ser instalado e montado para aplicação real. Só após este passo se pode realmente finalizar o objetivo inicial que consistia na construção de um

produto final com capacidades para ser comercializado e aplicado em situações reais com segurança e robustez.

Sendo todo o projeto desenvolvido no sentido de aplicação em meios industriais é importante que a caixa envolvente do equipamento seja resistente o suficiente para aguentar condições que podem não ser as mais benéficas para a sua. No entanto a garantia de durabilidade e tempo médio de bom funcionamento do equipamento tem obrigatoriamente de representar o principal objetivo durante a construção.

Começando por descrever as diferentes fases de evolução na construção do equipamento. A primeira fase na construção consiste em encontrar um método que permita aos utilizadores a ligação direta dos sensores às entradas do equipamento o que implica para isso a construção de uma placa onde serão incluídas as resistências que efetuarão a conversão dos sinais de corrente à entrada em sinais de tensão para as entradas analógicas do sistema. Uma vez que o limite máximo de tensão que pode ser analisado por uma porta analógica é 5V foi montado um circuito com resistências de 250Ω para que se desse a conversão de 4-20mA em 1-5V.

Na ligação ao Arduíno foram utilizadas todas as entradas analógicas enquanto para as saídas digitais foram escolhidas as portas 2,3 e 4 uma vez que as saídas 0 e 1 da placa de Arduíno são já utilizadas pelo funcionamento da placa de Ethernet. Construída e especificada toda esta base eletrónica da placa procedeu-se à fase seguinte de construção dos pinos através dos quais será efetuada a ligação ao Arduíno e aos pontos de ligação da placa com os conetores que se irão situar na zona exterior da caixa. Estes pontos de ligação serão os responsáveis pela transmissão do sinal entre os sensores exteriores e as entradas e saídas do Arduíno. Depois de efetuados todos estes processos de construção a placa final obtida é a apresentada na Figura 26.

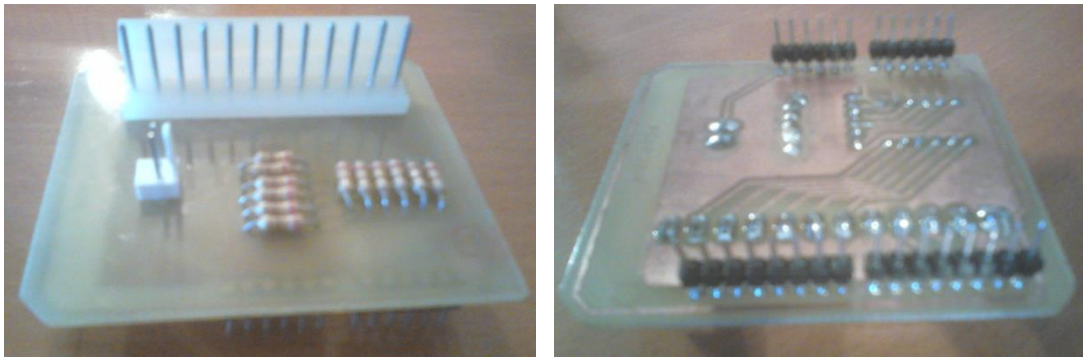

Figura 26: A imagem da esquerda apresenta a parte superior da placa e a imagem da direita a parte inferior

A fase seguinte consistiu na montagem dos conetores exteriores onde será efetuada a ligação de todos os sensores e de onde serão efetuadas as ligações dos pontos das saídas digitais.

A especificidade deste conetor passa pela correspondência a cada uma das saídas ou entradas digitais aos conetores que terão a sempre a ligação à referência (“terra”) posicionada na posição imediatamente a seguir. Assim no final a montagem ficou com a apresentação mostrada na Figura 27.

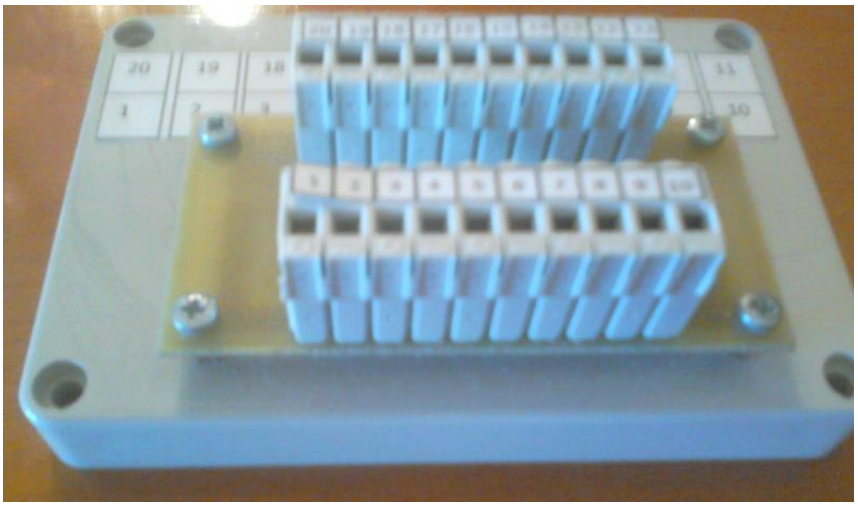

Figura 27: Montagem dos conetores na caixa do produto

O esquema de montagem que os utilizadores poderão usar como guia de referência informativo da posição na qual deverão ser ligados os sensores e de onde irão sair os sinais digitais está apresentado na Figura 28, e é também apresentado no exterior do equipamento.

e+ I&D FCUP/EWEN Arduino Project									
1	2	3	4	5	6	7	8	9	10
AI 0	AGND	AI 1	AGND	AI 2	AGND	AI 3	AGND	AI 4	AGND
11	12	13	14	15	16	17	18	19	20
AGND	AI 5			DGND	DO 2	DGND	DO 1	DGND	DO 0

Figura 28: Descrição das ligações aos conetores

Após o decorrer natural de todos estes processos para finalizar apenas resta a construção da caixa exterior e a abertura dos restantes furos. Foi acrescentada à caixa

um botão de reiniciar para recorrer em situações que exista um bloqueio ou um erro que altere o normal funcionamento do equipamento.

Encontrada a caixa para envolver o equipamento foi necessária a montagem do mesmo no interior desta, como podemos verificar na Figura 29, em que já todos os componentes se encontram nas respetivas posições finais.

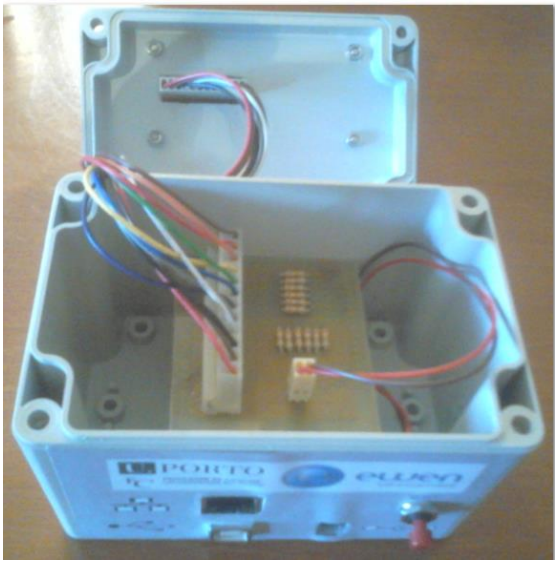

Figura 29: Imagem do interior da montagem final

Efetuada toda a montagem resta agora apenas fechar e selar a caixa e fica finalizada a apresentação do produto tal como pode ser visualizado na Figura 30.

Figura 30: Produto final

Está assim preparado o equipamento de deslastre e pronto para que surjam as primeiras oportunidades de aplicação em situações de controlo reais. Podem agora ser iniciados os primeiros testes e aprovadas as capacidades de autonomia e controlo do

equipamento. Desta forma o equipamento será capaz de operar sozinho desde que tenha energia, o que o torna numa ferramenta de longa duração sem necessidade de introdução de informação por parte dos utilizadores.

É possível após todo o processo de estudo e seleção usufruir por fim de um novo, simples e eficiente produto de deslastre.

Desenvolvimento do contador de impulsos

Descrito o projeto inicial o que se pretende é basicamente o desenvolvimento de um simples contador de impulsos mas com a particularidade de envio dos dados para um servidor para que possam ser desta forma monitorizados consumos de equipamentos ou instalações em tempo real. Para desenvolver estas capacidades o equipamento terá de permitir ao utilizador a possibilidade de atribuição das credenciais do servidor para onde pretende que sejam armazenados os dados, assim como a programação do período de integração com que pretende que estes dados sejam enviados para o servidor. Para além destas capacidades este equipamento deverá também permitir a ativação e desativação das entradas onde forem conectados os sensores que enviam os impulsos e onde poderá também ser possível a reiniciação desses mesmos contadores.

Colocadas as primeiras diretivas e já com algum conhecimento em desenvolvimento de tecnologia baseada em placas controladoras de Arduíno dei início então aos primeiros passos no sentido de resolver este caso.

Descrição da construção do código

Numa fase inicial comecei por desenvolver a capacidade de reconhecimento na leitura de impulsos recorrendo às capacidades das portas digitais contidas no Arduíno. Para isso criei contadores individuais que são incrementados sempre que à entrada é detetada uma passagem do valor 0 para o valor 1 e um regresso ao valor 0, ou seja é detetado um impulso.

O passo seguinte consistiu na criação de uma rotina para o envio dos dados das contagens para o servidor. Esta função foi desenvolvida recorrendo a uma biblioteca chamada SimpleTimer que tem como função a criação de uma interrupção que é independente do seguimento normal do código. Desta forma garante-se que não

existem perdas de dados de leituras sempre que é executada a função, pois não sabemos à partida o tempo mínimo de duração da mesma e portanto devemos assumir que eventualmente com o tempo vão ser perdidos dados de contagens que iriam coincidir com os momentos em que estavam a ser enviados os valores para o servidor. Isto tudo acontece porque esta linguagem executa as funções sequencialmente avançando para a próxima apenas depois da anterior ser completamente resolvida.

Ao equipamento será também ligada uma placa de RTC que será responsável pelo fornecimento da hora a que foi registado o valor da contagem para permitir aos utilizadores uma análise dos consumos correspondentes às contagens em função do tempo. Esta informação é crucial para que seja possível uma análise real e construtiva dos dados que estão a ser registados pelo equipamento ao longo do tempo. A particularidade deste RTC é que possui uma pilha que permite que mesmo em caso de falhas de energia ao equipamento o tempo continua a contar sem falhas, desta forma quando volta a energia conseguimos que a informação do tempo continue precisa e exata.

Criados os métodos básicos que garantem o bom funcionamento do sistema e tentam prever ao máximo a ocorrência de todo o tipo de falhas o que fiz foi recorrer ao servidor web para criar uma plataforma que permite a programação do equipamento. Nesta plataforma foram criados campos de configuração dos dados do servidor (IP, GateWay, DNS, Porta) na página de introdução do equipamento, como apresentado na Figura 31, constituindo assim um item obrigatório de preenchimento. Avançando esta etapa o campo que se segue é a apresentação de uma caixa obrigatória para atribuição do tempo de integração do equipamento, como o da Figura 32, sendo este também um campo obrigatório.

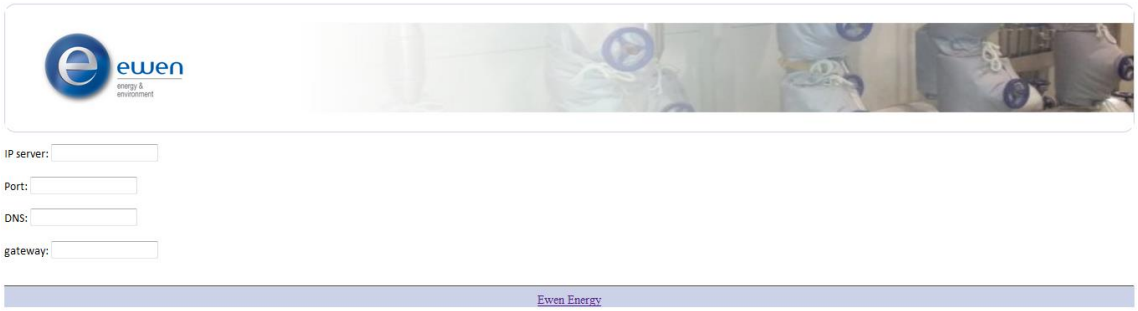

Figura 31: Definição dos dados do servidor

Depois de ultrapassados os campos de preenchimento obrigatório está disponível a página final onde são apresentadas as configurações gerais de atribuição dos respetivos canais aos contadores que se pretendem que sejam medidos. Nesta página será

mostrada informação sobre o valor atualizado a contagem de cada um dos contadores, e configurações tais como a possibilidade de ativação, desativação e reiniciar dos valores de cada um dos contadores.

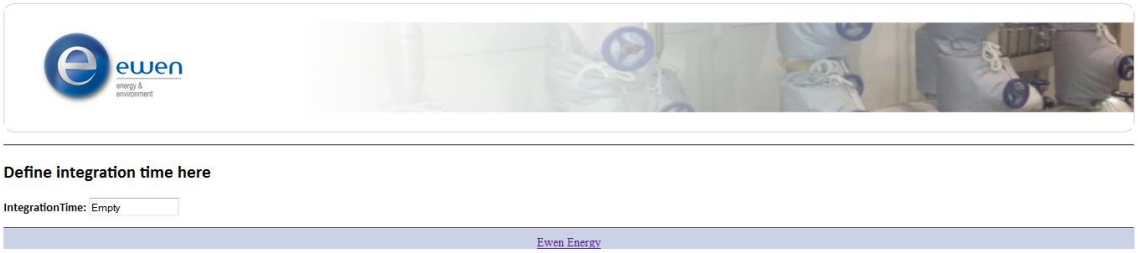

Figura 32: Definição do tempo de integração

Por fim foi efetuada a construção de uma rotina que funciona como cliente web e será responsável pelo envio dos dados para o servidor. Esta irá enviar os dados em formato "POST" que contém na mensagem uma máscara que indica quais as entradas que estão ativadas e a receber dados, a hora em que foi retirado esse registo dos valores e os valores dos respetivos contadores. Toda esta mensagem é enviada respeitando o formato JSON característico para o envio de dados através da web.

Elaborado o código resta apenas a fase final que consiste na escrita de um código que ative um servidor capaz de receber e interpretar os dados e executar o processo de armazenamento dos mesmos numa base de dados MYSQL instalada na máquina onde está presente no servidor. Depois de algum estudo e análise das linguagens que permitem a construção deste tipo de código foi evidente a escolha da criação de um programa em node.js que se trata de uma linguagem relativamente recente mas muito útil e direcionada para a execução deste tipo de funções.

Introdução à linguagem node.js

Node.js é uma linguagem escrita com base na máquina virtual Google V8 JavaScript. Enquanto os mecanismos de JavaScript são tradicionalmente executados em web Browsers para criarem o lado do cliente em aplicações cliente/servidor as bibliotecas de Node.js estão direcionadas para a construção de aplicações em JavaScript do lado do servidor.

As aplicações de Node.js são pretendidas para a aplicação em servidores dedicados de HTTP e são caracterizados por serem baseados em eventos e pela execução das suas funções de forma assíncrona. Este novo tipo de abordagem destaca-se desta forma do

tradicional tipo de abordagem em que o servidor recebe, processa, envia, espera e recebe antes que avance para o processo seguinte. Recorrendo a uma abordagem baseada em Node.js são executados múltiplos pedidos ao mesmo tempo não exigindo que o pedido anterior seja completamente executado antes de passar para o pedido seguinte. Desta forma o criador da linguagem (Ryan Dahl) garante-nos que com uma aplicação Node.js não ocorrem bloqueios entre as entradas e saídas. (TechTarget, 2014)

Para uma melhor perceção da simplicidade na criação de servidores web usando uma abordagem Node.js podemos observar a Figura 33 onde está apresentado um exemplo simples de criação de um servidor HTTP.

```
// Load the http module to create an http server.
var http = require('http');

// Configure our HTTP server to respond with Hello World to all requests.
var server = http.createServer(function (request, response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.end("Hello World\n");
});

// Listen on port 8000, IP defaults to 127.0.0.1
server.listen(8000);
```

Figura 33: Node.js exemplo de servidor HTTP (Caswell, 2014)

Se o tipo de servidor pretendido se tratar de um servidor TCP a criação de um deste tipo é também simples como está apresentado na Figura 34.

```
// Load the net module to create a tcp server.
var net = require('net');

// Creates a new TCP server. The handler argument is automatically set as a listener for the 'connection' event
var server = net.createServer(function (socket) {

  // Every time someone connects, tell them hello and then close the connection.
  console.log("Connection from " + socket.remoteAddress);
  socket.end("Hello World\n");

});

// Fire up the server bound to port 7000 on localhost
server.listen(7000, "localhost");
```

Figura 34: Node.js exemplo de um servidor TCP (Caswell, 2014)

Servidor para contador de impulsos

Para finalizar o projeto o último passo a seguir será a recolha e armazenamento dos dados que são periodicamente enviados para o servidor. Para tal, recorrendo a uma aplicação Node.js foi criado um servidor que está continuamente à espera de informação numa porta específica. Assim o importante será a construção de uma rotina capaz de interpretar a informação enviada pelo Arduíno e colocá-la de forma organizada em tabelas de MYQSL para análise posterior desses mesmos dados. Podemos agora dar inicio à recolha de informação.

Criada a aplicação que executa o transporte da informação para as bases de dados está então finalizado todo um processo que recolhe registos de parâmetros físicos desde uma instalação industrial para posterior análise. É este tipo de sistema que está na base de criação de plataformas de monitorização energética de instalações ou processos industriais.

No final o equipamento permitirá ao utilizador a atribuição de um nome e um conjunto de opções que permitem a reconfiguração dos dados relativos ao servidor e do tempo de integração com que são enviadas as contagens. De uma forma geral a apresentação final onde poderão ser visualizadas as informações do contador é a mostrada na Figura 35.

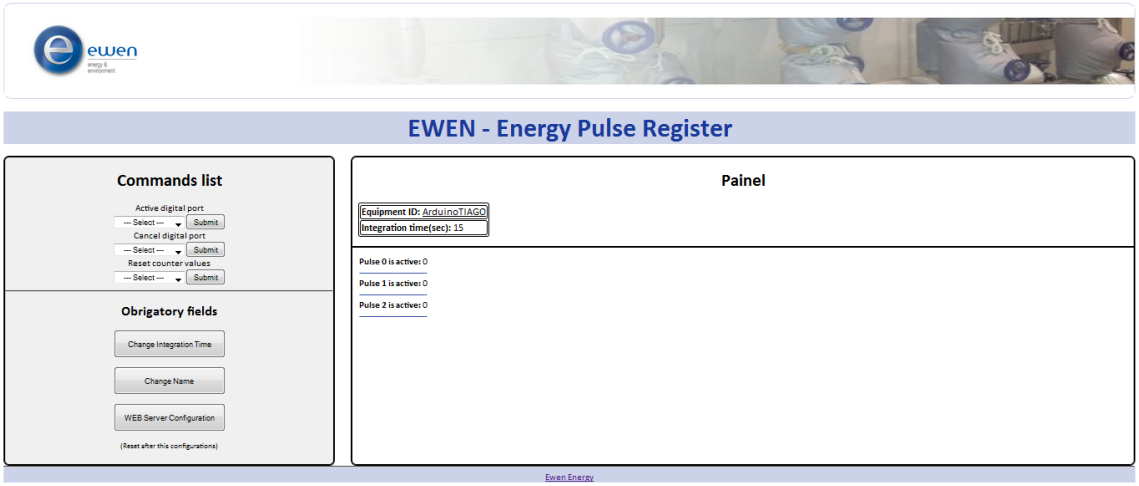

Figura 35: Página de configuração final do contador de impulsos

Construção da caixa

Elaborado todo o processo de desenvolvimento da tecnologia, estamos agora preparados para a construção da componente física que irá envolver o equipamento e prepará-lo para que seja possível a sua montagem numa situação real.

Uma vez que para o projeto anterior de deslastre foi já montada uma caixa, poderemos então usufruir da mesma onde a única diferença será a construção de uma nova placa que irá estabelecer a conexão entre o Arduíno e os conetores digitais. Devido ao uso da placa de Ethernet e do RTC ("*Real Time Clock*") apenas estarão disponíveis pelo Arduino as entradas digitais 2-10 sendo as restantes ocupadas para execução eficiente dos módulos anteriores. Após montada a placa a configuração de ligação dos contadores passará a ser a apresentada pela Figura 36.

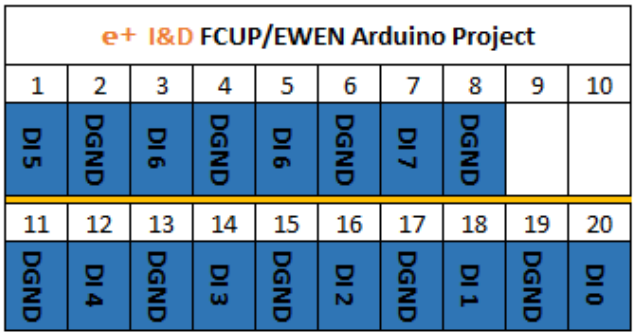

Figura 36: Representação das posições de ligação

Sendo que toda a restante montagem, inclusive, a numeração dos pins é a mesma apresentada pelo projeto de deslastre. Estamos assim preparados para usufruir desta nova tecnologia.

Conclusões

O Arduíno é um equipamento bastante versátil e simples para uso em aplicações com leitura de parâmetros analógicos para o qual existe um conjunto de bibliotecas que proporcionam uma rápida aprendizagem sobre o equipamento. No entanto é uma ferramenta direcionada para o desenvolvimento de projetos de reduzida complexidade o que implica limitações no espaço de memória e capacidade de processamento de dados.

O sistema de deslastre construído apresenta características particulares que fazem dele uma ferramenta simples mas com versatilidade para o controlo e monitorização de parâmetros físicos mostrando eficiência na apresentação de resultados. É um equipamento configurável a partir de um *browser* que pode ser acedido dentro da rede de cada instalação permitindo a alteração ou definição das suas funcionalidades sem a necessidade de deslocação até ao mesmo.

É um produto construído para apresentar enorme versatilidade na adaptação ao mais alargado número de condições ou imposições que se pretenda desenvolver, apresentando limitações na definição de condições temporais.

Após a aquisição de um *know-how* com o desenvolvimento do projeto de deslastre a adaptação e construção de um equipamento para contagem de impulsos revelou-se bastante mais rápida e eficaz. O contador apresenta características muito simplificadas apenas possuindo capacidades de leitura de dados por impulsos e envio simples sem rotinas para armazenamento dos mesmos. Esta caraterística transforma-o num equipamento com limitações no envio exato de todos os registos, e uma vez enviados durante uma falha de comunicação esses mesmos dados serão perdidos sem possibilidade de recuperação.

Para um bom desenvolvimento de ambos os projetos esteve sempre presente a construção organizada do código por forma a permitir no final obter o maior número de funcionalidades ocupando o menor espaço possível, sendo no entanto que todo o espaço livre de memória foi utilizado em ambos os projetos.

Trabalho futuro

Este trabalho representa um primeiro passo no sentido do desenvolvimento de um produto com as capacidades exigidas pelo mercado atual ultrapassando as limitações presentes na construção deste projeto inicial, onde será importante melhorar limitações de memória do equipamento em comparação com o utilizado neste projeto.

Uma das visões de trabalho futuro no projeto de deslastre consiste na implementação de um relógio que permita a definição de condições de tempo junto com as já implementadas. Acrescentando estas novas funções poderemos construir condições de tempo como o ligar ou desligar de sensores em períodos específicos de tempo ou então a implementação de ações que serão acionadas apenas quando uma condição se verifique ao final de um determinado período.

Ainda no projeto de deslastre será importante num futuro próximo desenvolver a capacidade de envio dos dados, que estão a ser controlados, para um servidor permitindo que um único equipamento se transforme num produto de monitorização e controlo de parâmetros. Esta nova capacidade permitiria não só controlar o bom ou mau funcionamento dos equipamentos assim como a implementação de condições de deslastre nesses mesmos sensores.

Uma área de estudo a desenvolver, ainda no equipamento de deslastre, será a construção de uma placa que permita utilização de diferentes tipos de sensores evitando a imposição obrigatória no uso de sensores de corrente. Com todos os desenvolvimentos, naturalmente pretende se o aumento do número de entradas analógicas e de saídas digitais proporcionando maior capacidade.

Um último passo no desenvolvimento do sistema seria o acréscimo de uma configuração de envio de uma mensagem por telemóvel ou correio eletrónico quando uma condição é verificada, de forma a transformar o equipamento numa ferramenta de monitorização mais útil para gestão industrial.

Pensando nas capacidades do contador de impulsos existe um trabalho de desenvolvimento importante na melhoria dos métodos de armazenamento será importante garantir que nunca existe perda de dados. Ou seja, quando existe uma falha na comunicação o equipamento terá capacidade de deteção da mesma onde consequentemente serão armazenados todos os valores da mesma. De seguida e quando detetar a abertura de comunicação de novo o equipamento enviará os dados

armazenados pela ordem com que foram guardados e apenas enviará os mais recentes quando forem corretamente enviados os antigos.

A fase seguinte será a transformação da comunicação de dados com o servidor para um formato de pergunta/resposta em que o equipamento envia a mensagem mas só enviará a próxima e marcará essa como enviada após a receção de uma resposta de "ok" enviada pelo lado do servidor a avisar que recebeu os dados com sucesso.

Bibliografia

Industrologic, Inc. (15 de 10 de 2014). Obtido de Using a 0-5 Volt DC Analog Input to Read Other Signal Levels: http://www.industrologic.com/aninput.htm

Acromag, I. (s.d.). *INTRODUCTION TO THE TWO-WIRE TRANSMITTER AND THE 4-20MA CURRENT LOOP.*

Arduino. (15 de 07 de 2014). *Compare board specs*. Obtido de arduino: http://arduino.cc/en/Products.Compare

arduinoclassroom. (12 de 08 de 2014). *Chapter 5: Analog Output.* Obtido de Arduino Classroom: http://www.arduinoclassroom.com/index.php/arduino-101/chapter-5

bapihvac. (10 de 08 de 2014). *Understanding 4-20 mA Current Loops.* Obtido de bapihvac: http://www.bapihvac.com/CatalogPDFs/I_App_Notes/Understanding_Current_Loops.pdf

Camera, D. (2013). *Using the EEPROM memory in AVR-GCC.* Dean Camera.

Caswell, T. (15 de 10 de 2014). *Hello Node!* Obtido de HOW TO NODE: http://howtonode.org/hello-node

Diffen. (25 de 5 de 2014). *Difference:TCP vs. UDP*. Obtido de diffen: http://www.diffen.com/difference/TCP_vs_UDP

ENERGÉTICOS, E. R. (2010). *ENTIDADE REGULADORA DOS SERVIÇOS ENERGÉTICOS.* Lisboa.

Feilipu. (12 de 06 de 2014). *"Goldilocks" 1284p Arduino UNO Clone*. Obtido de feilipu.me: http://feilipu.me/2013/03/08/goldilocks-1284p-arduino-uno-clone/

kickstarter. (12 de 06 de 2014). *Projects:DigiX - The ultimate Arduino compatible board with WiFi!* Obtido de kickstarter: https://www.kickstarter.com/projects/digistump/digix-the-ultimate-arduino-compatible-board-with-w

Ladyada. (16 de 07 de 2014). *Arduino Tutorial*. Obtido de ladyada: http://www.ladyada.net/learn/arduino/lesson4.html

Make:. (12 de 06 de 2014). *10 Hot New Boards to Watch*. Obtido de makezine: http://makezine.com/2013/06/10/10-hot-new-boards-to-watch/

Make:. (12 de 06 de 2014). *BeagleBone Black Has Arrived*. Obtido de makezine: http://makezine.com/2013/04/22/beaglebone-black-has-arrived/

Pohlmann, K. (1995). *Principles of Digital Audio.* McGraw-Hill.

Romeral, C. (16 de 07 de 2014). *RAM issues in Arduino atmega328p*. Obtido de Learning Purposeful Science: http://purposefulscience.blogspot.pt/2013/06/saving-ram-in-arduino-to-fit-sd-library.html

Schuman, L. (12 de 05 de 2014). *make:electronics:The Arduino Platform(ATmega88PA, ATmega168PA, and ATmega328P).* Obtido de Lars Schuman: http://make.larsi.org/electronics/ATmegaX8/

SparkFun Electronics [US]. (12 de 06 de 2014). *products: pcDuino2 - Dev Board*. Obtido de sparkfun: https://www.sparkfun.com/products/12749

TechTarget. (15 de 10 de 2014). *Node.js*. Obtido de Whatls.com: http://whatis.techtarget.com/definition/Nodejs

UDOO. (12 de 06 de 2014). *270k!!! 1000% of the GOAL reached!* Obtido de UDOO: http://www.udoo.org/270k-1000-of-the-goal-reached/

wikibooks. (25 de 5 de 2014). *Communication Networks/TCP and UDP Protocols*. Obtido de wikibooks: http://en.wikibooks.org/wiki/Communication_Networks/TCP_and_UDP_Protocols